BRM1553D Software API Reference

MIL-STD-1553 Bus Controller, Remote Terminal and Bus Monitor Software Application Programming Interface

April, 2020

Sital Technology Ltd. 17 Atir Yeda St., Kfar-Saba, 4464313, ISRAEL
http://www.sitaltech.com

Table of Contents

stld1553.cpp File Reference

```
#include "stdafx.h"
#include <stdio.h>
#include <string.h>
#include "OperatingSystemDependencies.h"
#include "InterProcessSemaphore.h"
#include "stld1553.h"
#include "stld1553_returnCodes.h"
#include "stld1553_driverInterface.h"
#include "stld1553_memoryManager.h"
#include "stld1553_internal.h"
```

Functions

S16BIT _DECL sitalStld1553_GetLibraryVersion (U16BIT *wpMajorVersion, U16BIT *wpMinorVersion, U16BIT *wpBuildNumber, U16BIT *wpRevisionNumber)

Get this library version numbers.

U16BIT _DECL sitalStld1553_GetShortLibraryVersion (void)

Get this library version numbers in a short form.

U16BIT _DECL sitalStld1553_GetShortCoreVersion (void)

Get the device driver interface library version numbers in a short form.

S16BIT _DECL sitalStld1553_ConfigurePrintings (BOOLEAN bIsConsoleScreen, sitalPrintingsLevelEnum pleBasePrintingsLevel)

Configure the printings to the console screen according to given parameters. This configuration affects this library as well as the driver-interface library. No printings are made if the current user application has no console screen. Only printings of a level greater/equal to given base printings level will be made.

U16BIT _DECL sitalDevice_Register_Read (S16BIT swDevice, U16BIT wRegisterAddress)

Read and return the current value of the register at given address for given device.

S16BIT _DECL sitalDevice_Register_Write (S16BIT swDevice, U16BIT wRegisterAddress, U16BIT wRegisterValue)

Write given value into the register at given address for given device.

S16BIT _DECL sitalDevice_Register_UpdateDevice (S16BIT swDevice, U16BIT wRegisterAddress)

Use the current image of the registers to actually update the registers of given device.

U16BIT _DECL sitalDevice_Memory_Read (S16BIT swDevice, U16BIT wDeviceMemoryAddress)

Read and return the current value of the device memory word at given address for given device.

S16BIT _DECL sitalDevice_Memory_Write (S16BIT swDevice, U16BIT wDeviceMemoryAddress, U16BIT wDeviceMemoryValue)

Write given value into the memory word at given address for given device.

S8BIT *_DECL sitalStld1553_GetBlockStatusWordErrorString (U16BIT wMode, U16BIT wBlockStatus)

Build and return a string in which the designated errors are textually reported. If no error is designated, a null string is returned.

S16BIT _DECL sitalStld1553_GetReturnCodeDescriberString (S16BIT swError, S8BIT *szpErrorString, U16BIT wMaximumStringLength)

Return a string that describes given return code.

S8BIT *_DECL sitalStld1553_GetMessageTypeString (U16BIT wMessageType)

Build and return a text string describing given message type.

S16BIT _DECL sitalBcStld1553_Command_GetCode (U16BIT *wpCommandWord, U16BIT wRtAddress, U16BIT wMessageDirection, U16BIT wSubaddressOrMode, U16BIT wWordCountOrModeCode)

Create a suitable IEEE-1553 command word based on given parameters.

S16BIT _DECL sitalBcStld1553_Command_Parse (U16BIT wCommandWord, U16BIT *wpRtAddress, U16BIT *wpMessageDirection, U16BIT *wpSubaddressOrMode, U16BIT *wpWordCountOrModeCode)

Parse given IEEE-1553 command word.

S16BIT _DECL sitalDevice_Initialize (S16BIT swDevice, U16BIT wAccess, U16BIT wMode, U32BIT dwSizeOfAllocatedMemory, U32BIT dwRegistersAddress, U32BIT dwMemoryAddress)

Initialize hardware & software resources (i.e., memory and register space) of given device for a given mode of operation. Access modes:

- Card memory: The card of target device is accessed using the device driver.
- Simulated memory: A 64K or 4K chunk of host memory is allocated and manipulated as if it were hardware memory. In this mode the user can produce a binary image file, but cannot actually run a frame. This mode isn't supported currently.
- User memory: Memory and register addresses are passed to the library. This mode isn't supported currently.

S16BIT _DECL sitalDevice_Free (S16BIT swDevice)

Reset and free given device.

S16BIT _DECL sitalDevice_CoreConfiguration_Get (S16BIT swDevice, U16BIT *wpCoreConfiguration)

Get the core configuration of given device.

S16BIT _DECL sitalDevice_TimeTag_Set (S16BIT swDevice, U16BIT wTimeTag)

Set the time tag register to given value.

S16BIT _DECL sitalDevice_TimeTag_Get (S16BIT swDevice, U16BIT *wpTimeTag)

Get the current value of the time tag register.

S16BIT _DECL sitalDevice_TimeTag_Reset (S16BIT swDevice)

Reset the time tag register.

S16BIT _DECL sitalDevice_TimeTag_SetResolution (S16BIT swDevice, U16BIT wTimeTagResolution)

Set the resolution of the time tag register.

S16BIT _DECL sitalDevice_OperationalStatisticsCollection_Configure (S16BIT swDevice, U16BIT bIsOperationalStatisticsCollected)

Configure whether operational statistics shall be collected for given device. These operational statistics include host buffer, stack, and GPQ fullness statistics.

S16BIT _DECL sitalDevice_Irq_GetMode (S16BIT swDevice, U16BIT *wpInterruptMode, U16BIT *wpAutoClear)

Get the currently configured interrupt-related behavior (i.e., type of signal and post-read auto-clearing of interrupt status) for given device.

S16BIT _DECL sitalDevice_Irq_Configure (S16BIT swDevice, U16BIT wInterruptMode, U16BIT wAutoClear)

Configure interrupt-related behavior (i.e., type of signal and post-read auto-clearing of interrupt status) for given device.

S16BIT _DECL sitalDevice_Irq_Manipulate (S16BIT swDevice, U16BIT bIsInterruptEnabled, U32BIT dwIrqMask, void(_DECL *funcpExternalIsr)(S16BIT swDevice, U32BIT dwIrqStatus))

Enable/disable interrupt service routine calls in case of given interrupts for given device.

S16BIT _DECL sitalDevice_SetAsynchronousMessagesIsr (S16BIT swDevice, void(_DECL *funcpAsynchronousMessagesIsr)(S16BIT swDevice, U16BIT wMinorFrameId))

Set the interrupt service routine to use with given device in case some high priority asynchronous message violates the time frame of a minor frame.

S16BIT _DECL sitalDevice_Clock_SetFrequency (S16BIT swDevice, U16BIT wClockFrequency)

Set the clock frequency of given device to be the given frequency.

S16BIT _DECL sitalDevice_Isq_ReadEntry (S16BIT swDevice, sitalIsqEntryStructure *iespIsqEntry)

Read the oldest yet unread entry of the interrupt status queue, and track ISQ overruns.

S16BIT _DECL sitalDevice_Isq_Clear (S16BIT swDevice)

Clear the ISQ, and reset the ISQ pointer REGISTER.

S16BIT _DECL sitalDevice_Isq_Configure (S16BIT swDevice, U16BIT bIsIsqEnabled)

Enable/disable the interrupt status queue.

S16BIT _DECL sitalDevice_SetResponseTimeout (S16BIT swDevice, U16BIT wResponseTimeout)

Set given response timeout for given device.

S16BIT _DECL sitalDevice_ConfigureWatchdogTimeout (S16BIT swDevice, U16BIT bIsWatchdogEnabled, U16BIT wWatchdogTimeout)

Configure the watchdog timeout for given device, that is, either enable and set it to given timeout, or disable it, as requested.

S16BIT _DECL sitalDevice_ConfigureDecoder (S16BIT swDevice, U16BIT wDecodedInput, U16BIT wExpendedXingOption)

Configure the Manchester-II decoder.

S16BIT _DECL sitalDevice_ConfigureRamParityCheck (S16BIT swDevice, U16BIT wRamParityCheckEnabler)

Enable/disable RAM parity checking for hardware containing 17-bit buffered RAM.

S16BIT _DECL sitalDevice_Test_Registers (S16BIT swDevice, sitalDeviceTestResultStructure *dtrspTestResult)

Reset given device and perform device registers test. This test consists of the following steps:

- Loop over a predetermined set of tested device registers, and with each tested register verify that:
  - It properly responds r/w operations.
  - It is reset upon device reset.
- Verify the correct operation of the time tag system.

S16BIT _DECL sitalDevice_Test_Memory (S16BIT swDevice, sitalDeviceTestResultStructure *dtrspTestResult, U16BIT wWrittenValue)

Reset given device and perform device memory test. Fill all the memory of given device with given value, and verify that the written values may be read back from the device.

S16BIT _DECL sitalDevice_Test_Protocol (S16BIT swDevice, sitalDeviceTestResultStructure *dtrspTestResult)

S16BIT _DECL sitalDevice_Test_Interrupts (S16BIT swDevice, sitalDeviceTestResultStructure *dtrspTestResult)

Reset given device and perform device interrupts test. This test consists of the following steps:

- Set the device time tag resolution to test mode: Set bits 7-9 of its Configuration Register #2 to 011(2).
- Set the device to use level type interrupts: Set bit 3 of its Configuration Register #2 to 1(2).
- Configure the device to issue time tag rollover interrupts: Set bit 6 of its Interrupt Mask Register #1 to 1(2).
- Assign an ISR to the device that will record the IRQ status of informed interrupts.
- Make the device generate a time tag rollover interrupt: Load its time tag register with a value of 0xFFFF, and then increment it to force a time tag rollover.
- Verify that the expected interrupt has indeed been tracked by the assigned ISR.

S16BIT _DECL sitalDevice_Test_Vectors (S16BIT swDevice, sitalDeviceTestResultStructure *dtrspTestResult, char *szpVectorsFilePath)

Reset given device and perform device vector test. All the test vectors that are available in given vector file will be retrieved one at a time, and applied to given device hardware. After applying all these vectors, registers and memory will be read to ensure the test of given vector group passed indeed.

S16BIT _DECL sitalBc_Initialize (S16BIT swDevice, U32BIT dwOptions)

Initialize given device as a BC in accordance with given initialization options. Release any past allocations of device memory.

S16BIT _DECL sitalBc_MessageRetryPolicy_Set (S16BIT swDevice, U16BIT wNumberOfRetries, U16BIT wFirstRetryBus, U16BIT wSecondRetryBus, U16BIT wReserved)

Configure the message retry policy of given BC device to given number of retries and given first and second chance bus to retry the message with.

S16BIT _DECL sitalBc_Start (S16BIT swDevice, S16BIT swFrameId, S32BIT sdwFrameCount)

Make given BC device start running given major frame for given number of times.

S16BIT _DECL sitalBc_Stop (S16BIT swDevice)

Stop given BC device from transmitting frames, either at the end of the current message or at the end of the current frame (what comes first). In case of a failure to do so, reset given device.

S16BIT _DECL sitalBc_GetActivationState (S16BIT swDevice, U16BIT *wpCurrentState)

Get the activation state, idle or busy, of given BC device.

S16BIT _DECL sitalBc_DataBlock_Create (S16BIT swDevice, S16BIT swDataBlockId, U16BIT wDataBlockSize, U16BIT *wapBuffer, U16BIT wBufferSize)

Create for given BC device a data block of given ID and size, and fill its allocated device memory with the contents of given buffer.

S16BIT _DECL sitalBc_DataBlock_Delete (S16BIT swDevice, S16BIT swDataBlockId)

Delete given data block which has been previously created for given BC device, and free the segment of device memory that has been allocated for it.

S16BIT _DECL sitalBc_DataBlock_Write (S16BIT swDevice, S16BIT swDataBlockId, U16BIT *wapBuffer, U16BIT wBufferSize, U16BIT wOffset)

Write the contents of given buffer in given data block of given BC device at given offset. For this purpose consider the data block as a cyclic buffer, i.e., continue the copy operation from its start if/when reaching its end.

S16BIT _DECL sitalBc_DataBlock_Read (S16BIT swDevice, S16BIT swDataBlockId, U16BIT *wapBuffer, U16BIT wBufferSize, U16BIT wOffset)

Read the contents of given data block of given BC device starting at given offset into given buffer. For this purpose consider the data block as a cyclic buffer, i.e., continue the copy operation from its start if/when reaching its end.

S16BIT _DECL sitalBc_Frame_Create (S16BIT swDevice, S16BIT swFrameId, U16BIT wFrameType, S16BIT *swapCommandIds, U16BIT wCommandCount, U16BIT wFrameTime, U16BIT wFlags)

Create for given BC device a new frame and calculate the overall number of included commands.

S16BIT _DECL sitalBc_Frame_Delete (S16BIT swDevice, S16BIT swFrameId)

Delete given frame which has been previously created for given BC device.

S16BIT _DECL sitalBc_Command_DistortWord (U16BIT wHardwareOpcode, U16BIT wDistortionManner, U16BIT wDistortionParameter)

Configure function sitalBc_Command_GetWord of the stld1553 library to insert from now on given distort to its generated commands.

S16BIT _DECL sitalBc_Command_GetWord (U16BIT wHardwareOpcode, U16BIT wCondition, U16BIT *wpCommandOpcode)

Generate a command word using given opcode and condition code. Odd parity is used for the generation of the parity bit (MSb) of the command opcode word.

S16BIT _DECL sitalBc_Command_Create (S16BIT swDevice, S16BIT swCommandId, U16BIT wOpcode, U16BIT wCondition, U32BIT dwParameter1, U32BIT dwParameter2, U32BIT dwReserved)

Create for given BC device a new command using given command descriptors. The new command will be actually allocated device memory space and written in it only at the right moments along the course of the life of its frame.

S16BIT _DECL sitalBc_Command_Delete (S16BIT swDevice, S16BIT swCommandId)

Delete given command which has been previously created for given BC device.

S16BIT _DECL sitalBc_Message_CreateBcToRt (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wReceiverRt, U16BIT wReceiverRtSubaddress, U16BIT wWordCount, U16BIT wGapTime, U32BIT dwOptions)

Create for given BC device a BC-to-RT message.

S16BIT _DECL sitalBc_Message_CreateRtToBc (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wTransmitterRt, U16BIT wTransmitterRtSubaddress, U16BIT wWordCount, U16BIT wGapTime, U32BIT dwOptions)

Create for given BC device a RT-to-BC message.

S16BIT _DECL sitalBc_Message_CreateRtToRt (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wReceiverRt, U16BIT wReceiverRtSubaddress, U16BIT wWordCount, U16BIT wTransmitterRt, U16BIT wTransmitterRtSubaddress, U16BIT wGapTime, U32BIT dwOptions)

Create for given BC device a RT-to-RT message.

S16BIT _DECL sitalBc_Message_CreateBcToOrFromRtMode (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wRt, U16BIT wMessageDirection, U16BIT wModeCommand, U16BIT wGapTime, U32BIT dwOptions)

Create for given BC device a BC-to-RT-Mode or a RT-to-BC-Mode message, depending on given RT device-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX, respectively).

S16BIT _DECL sitalBc_Message_CreateBcToBroadcast (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wReceiverRtSubaddress, U16BIT wWordCount, U16BIT wGapTime, U32BIT dwOptions)

Create for given BC device a BC-to-Broadcast message.

S16BIT _DECL sitalBc_Message_CreateRtToBroadcast (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wReceiverRtSubaddress, U16BIT wWordCount, U16BIT wTransmitterRt, U16BIT wTransmitterRtSubaddress, U16BIT wGapTime, U32BIT dwOptions)

Create for given BC device a RT-to-Broadcast message.

S16BIT _DECL sitalBc_Message_CreateBcToOrFromBroadcastMode (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wMessageDirection, U16BIT wModeCommand, U16BIT wGapTime, U32BIT dwOptions)

Create for given BC device a BC-to-Broadcast-Mode or a Broadcast-to-BC-Mode message, depending on given RT device-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX, respectively).

S16BIT _DECL sitalBc_Message_Create (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockIdForMessage1, U16BIT wBcControlWordForMessage1, U16BIT wCommandWord1ForMessage1, U16BIT wCommandWord2ForMessage1, U16BIT wGapTimeForMessage1, S16BIT swDataBlockIdForMessage2, U16BIT wBcControlWordForMessage2, U16BIT wCommandWord1ForMessage2, U16BIT wCommandWord2ForMessage2, U16BIT wGapTimeForMessage2, U32BIT dwOptions)

Create for given BC device a single/dual message for future use by some command, and write it at a dedicated place in device memory.

S16BIT _DECL sitalBc_Message_ModifyBcToRt (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wReceiverRt, U16BIT wReceiverRtSubaddress, U16BIT wWordCount, U16BIT wGapTime, U32BIT dwOptions, U16BIT wModificationFlags)

Modify a BC-to-RT message that has been previously created for given BC device.

S16BIT _DECL sitalBc_Message_ModifyRtToBc (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wTransmitterRt, U16BIT wTransmitterRtSubaddress, U16BIT wWordCount, U16BIT wGapTime, U32BIT dwOptions, U16BIT wModificationFlags)

Modify a RT-to-BC message that has been previously created for given BC device.

S16BIT _DECL sitalBc_Message_ModifyRtToRt (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wReceiverRt, U16BIT wReceiverRtSubaddress, U16BIT wWordCount, U16BIT wTransmitterRt, U16BIT wTransmitterRtSubaddress, U16BIT wGapTime, U32BIT dwOptions, U16BIT wModificationFlags)

Modify a RT-to-RT message that has been previously created for given BC device.

S16BIT _DECL sitalBc_Message_ModifyBcToOrFromRtMode (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wRt, U16BIT wMessageDirection, U16BIT wModeCommand, U16BIT wGapTime, U32BIT dwOptions, U16BIT wModificationFlags)

Modify a BC-to-RT-Mode or RT-to-BC message that has been previously created for given BC device, depending on given RT device-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX, respectively).

S16BIT _DECL sitalBc_Message_ModifyBcToBroadcast (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wReceiverRtSubaddress, U16BIT wWordCount, U16BIT wGapTime, U32BIT dwOptions, U16BIT wModificationFlags)

Modify a BC-to-Broadcast message that has been previously created for given BC device.

S16BIT _DECL sitalBc_Message_ModifyRtToBroadcast (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wReceiverRtSubaddress, U16BIT wWordCount, U16BIT wTransmitterRt, U16BIT wTransmitterRtSubaddress, U16BIT wGapTime, U32BIT dwOptions, U16BIT wModificationFlags)

Modify a RT-to-Broadcast message that has been previously created for given BC device.

S16BIT _DECL sitalBc_Message_ModifyBcToOrFromBroadcastMode (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wMessageDirection, U16BIT wModeCommand, U16BIT wGapTime, U32BIT dwOptions, U16BIT wModificationFlags)

Modify a BC-to-Broadcast-Mode or Broadcast-to-BC-Mode message that has been previously created for given BC device, depending on given RT device-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX, respectively).

S16BIT _DECL sitalBc_Message_Modify (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockIdForMessage1, U16BIT wBcControlWordForMessage1, U16BIT wCommandWord1ForMessage1, U16BIT wCommandWord2ForMessage1, U16BIT wGapTimeForMessage1, S16BIT swDataBlockIdForMessage2, U16BIT wBcControlWordForMessage2, U16BIT wCommandWord1ForMessage2, U16BIT wCommandWord2ForMessage2, U16BIT wGapTimeForMessage2, U16BIT wModificationFlags)

Modify a message that has been previously created for given BC device.

S16BIT _DECL sitalBc_Message_Delete (S16BIT swDevice, S16BIT swMessageId)

Delete given message which has been previously created for given BC device.

S16BIT _DECL sitalBc_Message_GetByIdRaw (S16BIT swDevice, S16BIT swMessageId, U16BIT *wapBuffer, U16BIT wIsPurgeRequired)

Read given message of given BC device in its raw state into given buffer, and then purge given message if so requested.

S16BIT _DECL sitalBc_Message_DecodeRaw (S16BIT swDevice, U16BIT *wapBuffer, sitalDecodedMessageStructure *dmspDecodedMessage)

Decode given message of given BC device into given structure.

S16BIT _DECL sitalBc_Message_GetByIdDecoded (S16BIT swDevice, S16BIT swMessageId, sitalDecodedMessageStructure *dmspDecodedMessage, U16BIT wIsPurgeRequired)

Read given message of given BC device, decode it into given structure, and then purge given message if so requested.

S16BIT _DECL sitalBc_ConfigureMessageGapTimeFieldPolicy (S16BIT swDevice, U16BIT bIsMessageGapTimeFieldEnabled)

Configure given BC device to enable/disable the message gap time field for all messages.

S16BIT _DECL sitalBc_Gpf_SetState (S16BIT swDevice, U16BIT wGpfNumber, U16BIT wEffect)

Set given effect with the state of given GFP condition of given BC device.

S16BIT _DECL sitalBc_GetConditionState (S16BIT swDevice, U16BIT wCondition, U16BIT *wpCurrentState)

Read the BC Condition Code register of given BC device and obtain the current state of given condition.

S16BIT _DECL sitalBc_Gpq_Read (S16BIT swDevice, sitalGpqEntryStructure *gespGpqEntry)

Read the next unread entry from the GPQ of given BC device.

S16BIT _DECL sitalBc_Gpq_GetEntriesCount (S16BIT swDevice)

Return the current number of available entries in the GPQ of given BC device.

S16BIT _DECL sitalBc_Gpq_HandleNewEntries (S16BIT swDevice)

Handle the new entries of the GPQ of given BC device. The way each GPQ entry is handled depends on its contents, and may include making some internal updates, calling the user application's ISR, and recording the latest minor frame that was transmitted by given BC device in its host buffer, if one is available.

S16BIT _DECL sitalBc_Gpq_GetOperationalStatistics (S16BIT swDevice, sitalGpqOperationalStatisticsStructure *gosspGpqOperationalStatistics, U16BIT bIsResetOfHighestRecordedPercentageRequired)

Return performance information about the general purpose queue of given BC device.

S16BIT _DECL sitalBc_AsynchronousMessage_CreateBcToRt (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wReceiverRt, U16BIT wReceiverRtSubaddress, U16BIT wWordCount, U16BIT wGapTime, U32BIT dwOptions, U16BIT *wapBuffer)

Create for given BC device:

- A double data block of given ID, and fill its allocated device memory with the contents of given buffer.
- An asynchronous message of type BC-to-RT that uses that data block.

S16BIT _DECL sitalBc_AsynchronousMessage_CreateRtToBc (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wTransmitterRt, U16BIT wTransmitterRtSubaddress, U16BIT wWordCount, U16BIT wGapTime, U32BIT dwOptions, U16BIT *wapBuffer)

Create for given BC device an asynchronous message of type RT-to-BC.

S16BIT _DECL sitalBc_AsynchronousMessage_CreateRtToRt (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wReceiverRt, U16BIT wReceiverRtSubaddress, U16BIT wWordCount, U16BIT wTransmitterRt, U16BIT wTransmitterRtSubaddress, U16BIT wGapTime, U32BIT dwOptions, U16BIT *wapBuffer)

Create for given BC device an asynchronous message of type RT-to-RT.

S16BIT _DECL sitalBc_AsynchronousMessage_CreateBcToOrFromRtMode (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wRt, U16BIT wMessageDirection, U16BIT wModeCommand, U16BIT wGapTime, U32BIT dwOptions, U16BIT *wapBuffer)

Create for given BC device an asynchronous message of type BC-to-RT-Mode or RT-to-BC-Mode, depending on given RT device-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX, respectively).

S16BIT _DECL sitalBc_AsynchronousMessage_CreateBcToBroadcast (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wReceiverRtSubaddress, U16BIT wWordCount, U16BIT wGapTime, U32BIT dwOptions, U16BIT *wapBuffer)

Create for given BC device an asynchronous message of type BC-to-Broadcast.

S16BIT _DECL sitalBc_AsynchronousMessage_CreateRtToBroadcast (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wReceiverRtSubaddress, U16BIT wWordCount, U16BIT wTransmitterRt, U16BIT wTransmitterRtSubaddress, U16BIT wGapTime, U32BIT dwOptions, U16BIT *wapBuffer)

Create for given BC device an asynchronous message of type RT-to-Broadcast.

S16BIT _DECL sitalBc_AsynchronousMessage_CreateBcToOrFromBroadcastMode (S16BIT swDevice, S16BIT swMessageId, S16BIT swDataBlockId, U16BIT wMessageDirection, U16BIT wModeCommand, U16BIT wGapTime, U32BIT dwOptions, U16BIT *wapBuffer)

Create for given BC device an asynchronous message of type BC-to-Broadcast-Mode or Broadcast-to-BC-Mode, depending on given RT device-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX, respectively).

S16BIT _DECL sitalBc_AsynchronousMessage_SendAtLowPriority (S16BIT swDevice, U16BIT *wpRemainingMessageCount, U16BIT wTimeFactor)

Fix for given BC device the subroutine that asynchronously transmits messages at low priority, so that it will transmit all the currently defined asynchronous messages. Then raise the proper general purpose flag, GPF-7, so that this subroutine will be actually called along the currently ongoing frame run.

S16BIT _DECL sitalBc_AsynchronousMessage_SendAtHighPriority (S16BIT swDevice, U16BIT wMessageId, U16BIT wTimeFactor)

Fix for given BC device the subroutine that asynchronously transmits a single desired message at high priority, so that it will transmit given message. Then raise the proper general purpose flag, GPF-6, so that this subroutine will be actually called along the currently ongoing frame run. Finally, wait for given message to be transmitted.

S16BIT _DECL sitalBc_HostBuffer_Initialize (S16BIT swDevice, U32BIT dwHostBufferSize)

Initialize (or re-initialize) given BC device's host buffer.

S16BIT _DECL sitalBc_HostBuffer_Free (S16BIT swDevice)

Free given bc device's host buffer.

S16BIT _DECL sitalBc_HostBuffer_Message_GetCount (S16BIT swDevice)

Get the number of messages that are currently available in the host buffer that is assigned with given bc device.

S16BIT _DECL sitalBc_HostBuffer_Message_GetDecoded (S16BIT swDevice, sitalDecodedMessageStructure *dmspDecodedMessage, U32BIT *dwpMessageCount, U32BIT *dwpLostMessageCount, U16BIT wMessageLocationAndRemoval)

Read from the host buffer of given BC device the message at given location, and decode it into given structure. Remove from the host buffer the message that was read.

S16BIT _DECL sitalBc_HostBuffer_Message_GetRaw (S16BIT swDevice, U16BIT *wapBuffer, U16BIT wBufferSize, U32BIT *dwpMessageCount, U32BIT *dwpLostMessageCount)

Read from the host buffer of given bc device as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the host buffer the messages that were read.

S16BIT _DECL sitalBc_HostBuffer_GetOperationalStatistics (S16BIT swDevice, sitalHostBufferOperationalStatisticsStructure *hbosspHostBufferOperationalStatistics, U16BIT bIsResetOfHighestRecordedPercentageRequired)

Return performance information about the host buffer of given BC device.

S16BIT _DECL sitalRt_Initialize (S16BIT swDevice, U16BIT wCommandStackSize, U32BIT dwOptions)

Initialize given device as a RT in accordance with given initialization options. Release any past allocations of device memory.

S16BIT
_DECL sitalRt_AddressSource_Set (S16BIT swDevice, U16BIT wRtAddressSource)

Set RT address source of given RT device to the one given.

S16BIT
_DECL sitalRt_AddressSource_Get (S16BIT swDevice, U16BIT *wpRtAddressSource)

Get RT address source of given RT device.

S16BIT
_DECL sitalRt_Address_Relatch (S16BIT swDevice)

Latch the RT address that is currently input to given RT device.

S16BIT
_DECL sitalRt_Address_Set (S16BIT swDevice, U16BIT wRtAddress)

Set the RT address of given RT device.

S16BIT
_DECL sitalRt_Address_Get (S16BIT swDevice, U16BIT *wpRtAddress)

Get the RT address of given RT device.

S16BIT
_DECL sitalRt_Start (S16BIT swDevice)

Make given RT device start receiving messages.

S16BIT
_DECL sitalRt_Stop (S16BIT swDevice)

Stop given RT device from responding to received messages.

S16BIT sitalRt_DataBlock_GetSize (U16BIT wDataBlockType, U16BIT
_DECL *wpSizeOfAllocatedDeviceMemory, U16BIT *wpActualSizeOfDataBlock)

Get the actual size of data block and the size it is allocated in device memory for given type of RT data block.

S16BIT sitalRt_DataBlock_Create (S16BIT swDevice, S16BIT swDataBlockId, U16BIT
_DECL wDataBlockType, U16BIT *wapBuffer, U16BIT wBufferSize)

Create for given RT device a data block of given ID and type, and fill it with the contents of given buffer. Use given type to determine both the size of the new data block and whether it will be used by some specific subaddress or globally. In case a common circular data block is created, enter it into use as the common circular buffer of given RT device.

S16BIT sitalRt_DataBlock_GetAddress (S16BIT swDevice, S16BIT swDataBlockId,
_DECL U16BIT *wpDeviceMemoryAddress)

Get the device memory address allocated for given data block of given RT device.

S16BIT
_DECL sitalRt_DataBlock_Delete (S16BIT swDevice, S16BIT swDataBlockId)

Delete given previously created data block of given RT device, and free the segment of device memory that has been allocated for it.

S16BIT _DECL sitalRt_DataBlock_Write (S16BIT swDevice, S16BIT swDataBlockId, U16BIT *wapBuffer, U16BIT wBufferSize, U16BIT wOffset)

Write the contents of given buffer in given data block of given RT device at given offset. For this purpose consider the data block as a cyclic buffer, i.e., continue the copy operation from its start if/when reaching its end.

S16BIT _DECL sitalRt_DataBlock_Read (S16BIT swDevice, S16BIT swDataBlockId, U16BIT *wapBuffer, U16BIT wBufferSize, U16BIT wOffset)

Read the contents of given data block of given RT device starting at given offset into given buffer. For this purpose consider the data block as a cyclic buffer, i.e., continue the copy operation from its start if/when reaching its end.

S16BIT _DECL sitalRt_DataBlock_MapToSubaddress (S16BIT swDevice, S16BIT swDataBlockId, U16BIT wSubaddress, U16BIT wMessageTypes, U16BIT wIrqOptions, U16BIT bIsSubaddressLegalizationRequested)

Map given data block with given subaddress of given RT device for given message types. Enable given interrupts with given subaddress. If required, legalize given message types for:

- All mode code messages
- All messages of given subaddress.

S16BIT _DECL sitalRt_DataBlock_UnmapFromSubaddress (S16BIT swDevice, S16BIT swDataBlockId, U16BIT wSubaddress, U16BIT wMessageTypes)

Unmap given data block with given subaddress of given RT device for given message types.

S16BIT _DECL sitalRt_ModeCode_EnableIrq (S16BIT swDevice, U16BIT wModeCodeType, U16BIT wModeCodeIrq)

Set given RT device to generate given interrupts upon reception of given mode codes messages.

S16BIT _DECL sitalRt_ModeCode_DisableIrq (S16BIT swDevice, U16BIT wModeCodeType, U16BIT wModeCodeIrq)

Set given RT device to stop the issue of given interrupts upon reception of given mode codes messages.

S16BIT _DECL sitalRt_ModeCode_GetIrq (S16BIT swDevice, U16BIT wModeCodeType, U16BIT *wpModeCodeIrq)

Let know whether given RT device is currently configured to generate any interrupts upon reception of given mode codes messages.

S16BIT _DECL sitalRt_ModeCode_ReadData (S16BIT swDevice, U16BIT wDataContainingModeCode, U16BIT *wpData)

Read the data for given mode code from the mode code data locations table of given RT device.

S16BIT _DECL sitalRt_ModeCode_WriteData (S16BIT swDevice, U16BIT wDataContainingModeCode, U16BIT wData)

Write given data for given mode code in the mode code data locations table of given RT device.

S16BIT _DECL sitalRt_MessageLegality_Enable (S16BIT swDevice, U16BIT wOwnAddressOrBroadcast, U16BIT wMessageDirection, U16BIT wSubaddress, U32BIT dwWordCountOrModeCodeMask)

Legalize messages received by given RT device if they suit given criteria.

S16BIT _DECL sitalRt_MessageLegality_Disable (S16BIT swDevice, U16BIT wOwnAddressOrBroadcast, U16BIT wMessageDirection, U16BIT wSubaddress, U32BIT dwWordCountOrModeCodeMask)

Illegalize messages received by given RT device if they suit given criteria.

S16BIT _DECL sitalRt_MessageLegality_GetStatus (S16BIT swDevice, U16BIT wOwnAddressOrBroadcast, U16BIT wMessageDirection, U16BIT wSubaddress, U32BIT *dwpMessageLegality)

Get the current legalization state of messages received by given RT device if they suit given criteria.

S16BIT _DECL sitalRt_MessageBusyBit_Set (S16BIT swDevice, U16BIT wOwnAddressOrBroadcast, U16BIT wMessageDirection, U32BIT dwSubaddressMask)

Make messages received by given RT device raise the busy bit in their status responses if they suit given criteria.

S16BIT _DECL sitalRt_MessageBusyBit_Clear (S16BIT swDevice, U16BIT wOwnAddressOrBroadcast, U16BIT wMessageDirection, U32BIT dwSubaddressMask)

Make messages received by given RT device not raise the busy bit in their status responses if they suit given criteria.

S16BIT _DECL sitalRt_MessageBusyBit_GetStatus (S16BIT swDevice, U16BIT wOwnAddressOrBroadcast, U16BIT wMessageDirection, U32BIT *dwpSubaddressMask)

Get the current busy bit state of messages received by given RT device if they suit given criteria.

S16BIT _DECL sitalRt_ResponseStatusBits_Set (S16BIT swDevice, U16BIT wStatusEnablerMask)

Configure given status enabler bits in order to control status responses made by given RT device. In case given RT device has been configured to be with/without

alternate status mode, given bits are set/cleared, respectively. This configuration of alternate status mode also affects the set of the relevant status bits.

S16BIT _DECL sitalRt_ResponseStatusBits_Unset (S16BIT swDevice, U16BIT wStatusEnablerMask)

Reverse configure given status enabler bits in order to control status responses made by given RT device. In case given RT device has been configured to be with/without alternate status mode, given bits are cleared/set, respectively. This configuration of alternate status mode also affects the set of the relevant status bits.

S16BIT _DECL sitalRt_ResponseStatusBits_Get (S16BIT swDevice, U16BIT *wpStatusEnablerMask)

Get the current configuration of status enabler bits, which are the configuration bits that control status responses made by given RT device. Whether given RT device is configured with/without alternate status mode affects the set of the relevant status bits.

S16BIT _DECL sitalRt_Message_GetFromStackRaw (S16BIT swDevice, U16BIT *wapBuffer, U16BIT wBufferSize)

Read from the RT stack of given RT device into given buffer as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the stack the messages that were read.

S16BIT _DECL sitalRt_Message_GetFromStackDecoded (S16BIT swDevice, sitalDecodedMessageStructure *dmspDecodedMessage, U16BIT wMessageLocationAndRemoval)

Read from the stack of given RT device the message at given location, decode it into given structure, and purge it if so required.

S16BIT _DECL sitalRt_Message_DecodeRaw (S16BIT swDevice, U16BIT *wapBuffer, sitalDecodedMessageStructure *dmspDecodedMessage)

Decode given message of given RT device into given structure.

S16BIT _DECL sitalRt_BuiltInTestWord_Configure (S16BIT swDevice, U16BIT wWordLocation, U16BIT wPermitOrInihibitIfRtBusy)

Configure the built-in test of given RT device to use the word in given source and to inhibit/permit the built-in test if the RT is busy, as required.

S16BIT _DECL sitalRt_BuiltInTestWord_Read (S16BIT swDevice, U16BIT wWordLocation, U16BIT *wpWord)

Read the built-in test word of given RT device that is stored at given location.

S16BIT _DECL sitalRt_BuiltInTestWord_Write (S16BIT swDevice, U16BIT wWord)

Write given built-in test word at the appropriate predefined memory address of given RT device.

S16BIT _DECL sitalRt_HostBuffer_Initialize (S16BIT swDevice, U32BIT dwHostBufferSize)

Initialize (or re-initialize) given RT device's host buffer.

S16BIT _DECL sitalRt_HostBuffer_Free (S16BIT swDevice)

Free given RT device's host buffer.

S16BIT _DECL sitalRt_HostBuffer_Message_Record (S16BIT swDevice)

Record the newly received messages from the command stack of given RT device into its host buffer.

S16BIT _DECL sitalRt_HostBuffer_Message_GetCount (S16BIT swDevice)

Get the number of messages that are currently available in the host buffer that is assigned with given RT device.

S16BIT _DECL sitalRt_HostBuffer_Message_GetDecoded (S16BIT swDevice, sitalDecodedMessageStructure *dmspDecodedMessage, U32BIT *dwpMessageCount, U32BIT *dwpStackLostMessageCount, U32BIT *dwpHostBufferLostMessageCount, U16BIT wMessageLocationAndRemoval)

Read from the host buffer of given RT device the message at given location, decode it into given structure, and purge it if so required. Also get the number of retrieved messages (actually only 0 or 1), the host buffer's current number of lost messages, and the current number of lost messages for given RT device's RT stack.

S16BIT _DECL sitalRt_HostBuffer_Message_GetRaw (S16BIT swDevice, U16BIT *wapBuffer, U16BIT wBufferSize, U32BIT *dwpMessageCount, U32BIT *dwpStackLostMessageCount, U32BIT *dwpHostBufferLostMessageCount)

Read from the host buffer of given RT device as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the host buffer the messages that were read.

S16BIT _DECL sitalRt_HostBuffer_GetOperationalStatistics (S16BIT swDevice, sitalHostBufferOperationalStatisticsStructure *hbosspHostBufferOperationalStatistics, U16BIT bIsResetOfHighestRecordedPercentageRequired)

Return performance information about the host buffer of given RT device.

S16BIT _DECL sitalMt_Initialize (S16BIT swDevice, U16BIT wStackMode, U16BIT wCommandStackSize, U16BIT wDataStackSize, U32BIT dwOptions)

Initialize given device as a MT in accordance with given initialization options. Release any past allocations of device memory.

S16BIT _DECL sitalMt_Stack_Swap (S16BIT swDevice)

Swap between the active and inactive stacks of given MT device.

S16BIT _DECL sitalMt_Stack_GetOperationalStatistics (S16BIT swDevice, sitalStackOperationalStatisticsStructure *sosspStackOperationalStatistics, U16BIT wStackSelector, U16BIT bIsResetOfHighestRecordedPercentageRequired)

Return performance information about the command stack of given MT device.

S16BIT _DECL sitalMt_GetInformation (S16BIT swDevice, sitalMtInformationStructure *mispMtInformation)

Return information about the configuration of given MT device.

S16BIT _DECL sitalMt_Start (S16BIT swDevice)

Make given MT device start capturing messages.

S16BIT _DECL sitalMt_Stop (S16BIT swDevice)

Make given MT device stop capturing messages.

S16BIT _DECL sitalMt_Pause (S16BIT swDevice)

Make given MT device temporarily stop capturing messages.

S16BIT _DECL sitalMt_Continue (S16BIT swDevice)

Make given MT device continue capturing messages after a temporary pause. The message capturing activity will continue from the same internal state as at the moment of pausing.

S16BIT _DECL sitalMt_MessageMonitoring_Enable (S16BIT swDevice, U16BIT wRtAddress, U16BIT wMessageDirection, U32BIT dwRtSubaddressMask)

Configure given MT device to monitor commands that suits given combinations of RT address, RT-related message direction, and subaddress.

S16BIT _DECL sitalHoo9_Mt_MessageMonitoring_Enable (S16BIT swDevice, U16BIT wRtAddress, U16BIT wMessageDirection, U64BIT qwRtSubaddressMask)

Configure given HOO9 MT device to monitor commands that suits given combinations of RT address, RT-related message direction, and subaddress.

S16BIT _DECL sitalPp194_Mt_MessageMonitoring_Enable (S16BIT swDevice, U16BIT wRtAddress, U16BIT wMessageDirection, U32BIT dwRtSubaddressMask)

Configure given PP194 MT device to monitor commands that suits given combinations of RT address, RT-related message direction/bus, and subaddress.

S16BIT _DECL sitalMt_MessageMonitoring_Disable (S16BIT swDevice, U16BIT wRtAddress, U16BIT wMessageDirection, U32BIT dwRtSubaddressMask)

Configure given MT device to avoid monitoring commands that suits given combinations of RT address, RT-related message direction/bus, and subaddress.

S16BIT _DECL sitalHoo9_Mt_MessageMonitoring_Disable (S16BIT swDevice, U16BIT wRtAddress, U16BIT wMessageDirection, U64BIT qwRtSubaddressMask)

Configure given HOO9 MT device to avoid monitoring commands that suits given combinations of RT address, RT-related message direction, and subaddress.

S16BIT _DECL sitalPp194_Mt_MessageMonitoring_Disable (S16BIT swDevice, U16BIT wRtAddress, U16BIT wMessageDirection, U32BIT dwRtSubaddressMask)

Configure given PP194 MT device to avoid monitoring commands that suits given combinations of RT address, RT-related message direction/bus, and subaddress.

S16BIT _DECL sitalMt_MessageMonitoring_GetStatus (S16BIT swDevice, U16BIT wRtAddress, U16BIT wMessageDirection, U32BIT *dwpRtSubaddressMask)

Get for given MT device and combination of RT address and RT-related message direction a mask specifying the monitored subaddresses.

S16BIT _DECL sitalHoo9_Mt_MessageMonitoring_GetStatus (S16BIT swDevice, U16BIT wRtAddress, U16BIT wMessageDirection, U64BIT *qwpRtSubaddressMask)

Get for given HOO9 MT device and combination of RT address and RT-related message direction a mask specifying the monitored subaddresses.

S16BIT _DECL sitalPp194_Mt_MessageMonitoring_GetStatus (S16BIT swDevice, U16BIT wRtAddress, U16BIT wMessageDirection, U32BIT *dwpRtSubaddressMask)

Get for given PP194 MT device and combination of RT address and RT-related message direction/bus a mask specifying the monitored subaddresses.

S16BIT _DECL sitalMt_Message_DecodeRaw (S16BIT swDevice, U16BIT *wapBuffer, sitalDecodedMessageStructure *dmspDecodedMessage)

Decode given message of given MT device into given structure.

S16BIT _DECL sitalHoo9_Mt_Message_DecodeRaw (S16BIT swDevice, U16BIT *wapBuffer, sitalDecodedMessageStructure *dmspDecodedMessage)

Decode given message of given HOO9 MT device into given structure.

S16BIT _DECL sitalPp194_Mt_Message_DecodeRaw (S16BIT swDevice, U16BIT *wapBuffer, sitalDecodedMessageStructure *dmspDecodedMessage)

Decode given message of given PP194 MT device into given structure.

S16BIT _DECL sitalMt_Message_GetFromStackRaw (S16BIT swDevice, U16BIT *wapBuffer, U16BIT wBufferSize, U16BIT wStackSelector)

Read from given stack of given MT device into given buffer as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the stack the messages that were read.

S16BIT _DECL sitalHoo9_Mt_Message_GetFromStackRaw (S16BIT swDevice, U16BIT *wapBuffer, U16BIT wBufferSize, U16BIT wStackSelector)

Read from given stack of given HOO9 MT device into given buffer as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the stack the messages that were read.

S16BIT _DECL sitalPp194_Mt_Message_GetFromStackRaw (S16BIT swDevice, U16BIT *wapBuffer, U16BIT wBufferSize, U16BIT wStackSelector)

Read from given stack of given PP194 MT device into given buffer as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the stack the messages that were read.

S16BIT _DECL sitalMt_Message_GetFromStackDecoded (S16BIT swDevice, sitalDecodedMessageStructure *dmspDecodedMessage, U16BIT wMessageLocationAndRemoval, U16BIT wStackSelector)

Read from given stack of given MT device the message at given location, decode it into given structure, and purge it if so required.

S16BIT _DECL sitalHoo9_Mt_Message_GetFromStackDecoded (S16BIT swDevice, sitalDecodedMessageStructure *dmspDecodedMessage, U16BIT wMessageLocationAndRemoval, U16BIT wStackSelector)

Read from given stack of given HOO9 MT device the message at given location, decode it into given structure, and purge it if so required.

S16BIT _DECL sitalPp194_Mt_Message_GetFromStackDecoded (S16BIT swDevice, sitalDecodedMessageStructure *dmspDecodedMessage, U16BIT wMessageLocationAndRemoval, U16BIT wStackSelector)

Read from given stack of given PP194 MT device the message at given location, decode it into given structure, and purge it if so required.

S16BIT _DECL sitalMt_HostBuffer_Initialize (S16BIT swDevice, U32BIT dwHostBufferSize)

Initialize (or re-initialize) given MT device's host buffer.

S16BIT _DECL sitalMt_HostBuffer_Free (S16BIT swDevice)

Free given MT device's host buffer.

S16BIT _DECL sitalMt_HostBuffer_Message_Record (S16BIT swDevice)

Record the newly received messages from the active command and data stacks of given MT device into its host buffer.

S16BIT _DECL sitalMt_HostBuffer_Message_GetCount (S16BIT swDevice)

Get the number of messages that are currently available in the host buffer that is assigned with given MT device.

S16BIT _DECL sitalMt_HostBuffer_Message_GetDecoded (S16BIT swDevice, sitalDecodedMessageStructure *dmspDecodedMessage, U32BIT *dwpMessageCount, U32BIT *dwpStackLostMessageCount, U32BIT *dwpHostBufferLostMessageCount, U16BIT wMessageLocationAndRemoval)

Read from the host buffer of given MT device the message at given location, decode it into given structure, and purge it if so required. Also get the number of retrieved messages (actually only 0 or 1), the host buffer's current number of lost messages, and the current number of lost messages for both given MT device's MT stacks.

S16BIT _DECL sitalHoo9_Mt_HostBuffer_Message_GetDecoded (S16BIT swDevice, sitalDecodedMessageStructure *dmspDecodedMessage, U32BIT *dwpMessageCount, U32BIT *dwpStackLostMessageCount, U32BIT *dwpHostBufferLostMessageCount, U16BIT wMessageLocationAndRemoval)

Read from the host buffer of given HOO9 MT device the message at given location, decode it into given structure, and purge it if so required. Also get the number of retrieved messages (actually only 0 or 1), the host buffer's current number of lost messages, and the current number of lost messages for both given MT device's MT stacks.

S16BIT _DECL sitalPp194_Mt_HostBuffer_Message_GetDecoded (S16BIT swDevice, sitalDecodedMessageStructure *dmspDecodedMessage, U32BIT *dwpMessageCount, U32BIT *dwpStackLostMessageCount, U32BIT *dwpHostBufferLostMessageCount, U16BIT wMessageLocationAndRemoval)

Read from the host buffer of given PP194 MT device the message at given location, decode it into given structure, and purge it if so required. Also get the number of retrieved messages (actually only 0 or 1), the host buffer's current number of lost messages, and the current number of lost messages for both given MT device's MT stacks.

S16BIT _DECL sitalMt_HostBuffer_Message_GetRaw (S16BIT swDevice, U16BIT *wapBuffer, U16BIT wBufferSize, U32BIT *dwpMessageCount, U32BIT *dwpStackLostMessageCount, U32BIT *dwpHostBufferLostMessageCount)

Read from the host buffer of given MT device as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the host buffer the messages that were read.

S16BIT _DECL sitalHoo9_Mt_HostBuffer_Message_GetRaw (S16BIT swDevice, U16BIT *wapBuffer, U16BIT wBufferSize, U32BIT *dwpMessageCount, U32BIT *dwpStackLostMessageCount, U32BIT *dwpHostBufferLostMessageCount)

Read from the host buffer of given HOO9 MT device as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the host buffer the messages that were read.

S16BIT _DECL sitalPp194_Mt_HostBuffer_Message_GetRaw (S16BIT swDevice, U16BIT *wapBuffer, U16BIT wBufferSize, U32BIT *dwpMessageCount, U32BIT *dwpStackLostMessageCount, U32BIT *dwpHostBufferLostMessageCount)

Read from the host buffer of given PP194 MT device as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the host buffer the messages that were read.

S16BIT _DECL sitalMt_HostBuffer_GetOperationalStatistics (S16BIT swDevice, sitalHostBufferOperationalStatisticsStructure *hbosspHostBufferOperationalStatistics, U16BIT bIsResetOfHighestRecordedPercentageRequired)

Return performance information about the host buffer of given MT device.

S16BIT _DECL sitalRtMt_Initialize (S16BIT swDevice, U16BIT wRtCommandStackSize, U16BIT wMtStackMode, U16BIT wMtCommandStackSize, U16BIT wMtDataStackSize, U32BIT dwOptions)

Initialize given device as a RT&MT in accordance with given initialization options. Release any past allocations of device memory.

S16BIT _DECL sitalRtMt_Start (S16BIT swDevice)

Make given RT&MT device start responding and capturing messages.

S16BIT _DECL sitalRtMt_Stop (S16BIT swDevice)

Stop given RT&MT device from responding to received messages.

S16BIT _DECL sitalRtMt_HostBuffer_Initialize (S16BIT swDevice, U32BIT dwHostBufferSize)

Initialize (or re-initialize) given device's RT&MT combined host buffer.

S16BIT _DECL sitalRtMt_HostBuffer_Free (S16BIT swDevice)

Free given device's RT&MT combined host buffer.

S16BIT _DECL sitalRtMt_HostBuffer_Message_Record (S16BIT swDevice)

Record the newly received messages from the active command and data stacks of given RT&MT device into its combined host buffer.

S16BIT _DECL sitalRtMt_HostBuffer_Message_GetCount (S16BIT swDevice)

Get the number of messages that are currently available in the combined host buffer that is assigned with given RT&MT device.

S16BIT _DECL sitalRtMt_HostBuffer_Message_GetDecoded (S16BIT swDevice, sitalDecodedMessageStructure *dmspDecodedMessage, U32BIT *dwpMessageCount, U32BIT *dwpRtStackLostMessageCount, U32BIT *dwpMtStackLostMessageCount, U32BIT *dwpHostBufferLostMessageCount, U16BIT wMessageLocationAndRemoval)

Read from the combined host buffer of given RT&MT device the message at given location, decode it into given structure, and purge it if so required. Also get the number of retrieved messages (actually only 0 or 1), the combined host buffer's current number of lost messages, the current number of lost messages for given

RT&MT device's RT stack, and the current number of lost messages for both given RT&MT device's MT stacks.

_EXTERN S16BIT _DECL sitalRtMt_HostBuffer_Message_GetRaw (S16BIT swDevice, U16BIT *wapBuffer, U16BIT wBufferSize, U32BIT *dwpMessageCount, U32BIT *dwpRtStackLostMessageCount, U32BIT *dwpMtStackLostMessageCount, U32BIT *dwpHostBufferLostMessageCount)

Read from the combined host buffer of given RT&MT device as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the combined host buffer the messages that were read.

S16BIT _DECL sitalRtMt_HostBuffer_GetOperationalStatistics (S16BIT swDevice, sitalHostBufferOperationalStatisticsStructure *hbosspHostBufferOperationalStatistics, U16BIT bIsResetOfHighestRecordedPercentageRequired)

Return performance information about the combined host buffer of given RT&MT device.

Function Documentation

S16BIT _DECL
sitalBc_AsynchronousMessage_CreateBcToBroadcast

```
( S16BIT   swDevice,
  S16BIT   swMessageId,
  S16BIT   swDataBlockId,
  U16BIT   wReceiverRtSubaddress,
  U16BIT   wWordCount,
  U16BIT   wGapTime,
  U32BIT   dwOptions,
  U16BIT * wapBuffer
)
```

Create for given BC device an asynchronous message of type BC-to-Broadcast.

Note:

- An asynchronous message is asynchronously transmitted at low priority in result of calling function sitalBc_AsynchronousMessage_SendAtLowPriority.
- An asynchronous message is asynchronously transmitted at high priority in result of calling function sitalBc_AsynchronousMessage_SendAtHighPriority.
- Unlike a normal message, an asynchronous message may not be transmitted by user-defined commands; This is prohibited by function sitalBc_Command_Create.

Equivalent DDC definition: aceBCAsyncMsgCreateBcst

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wReceiverRtSubaddress	(in) Receiver RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wWordCount	(in) Word count (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)

dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)
wapBuffer	(in) A pointer to an array of data words to be copied into the new data block

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_DataBlock_Create, sitalBc_Message_CreateBcToBroadcast, sitalBcControlWord_BUS_A, sitalBcControlWord_LOOPBACK_TEST_PERFORMED, sitalBcDataBlockSizeOption_DOUBLE, sitalBcMessageOption_DUAL_MESSAGE, sitalBcSetupOption_ASYNCHRONOUS_BOTH_PRIORITY_MODES, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_ASYNCHRONOUS_MESSAGE_ERROR, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalReturnCode_SUITABLE_ASYNCHRONOUS_MODE_UNDEFINED.

S16BIT _DECL **sitalBc_AsynchronousMessage_CreateBcToOrFromBroadcastMode** (
S16BIT *swDevice*,
S16BIT *swMessageId*,
S16BIT *swDataBlockId*,
U16BIT *wMessageDirection*,
U16BIT *wModeCommand*,
U16BIT *wGapTime*,
U32BIT *dwOptions*,
U16BIT * *wapBuffer*
)

Create for given BC device an asynchronous message of type BC-to-Broadcast-Mode or Broadcast-to-BC-Mode, depending on given RT device-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX, respectively).

Note:

- An asynchronous message is asynchronously transmitted at low priority in result of calling function sitalBc_AsynchronousMessage_SendAtLowPriority.
- An asynchronous message is asynchronously transmitted at high priority in result of calling function sitalBc_AsynchronousMessage_SendAtHighPriority.
- Unlike a normal message, an asynchronous message may not be transmitted by user-defined commands; This is prohibited by function sitalBc_Command_Create.

Equivalent DDC definition: aceBCAsyncMsgCreateBcstMode

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))

swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX)
wModeCommand	(in) Mode command (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)
wapBuffer	(in) A pointer to an array of data words to be copied into the new data block

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_DataBlock_Create, sitalBc_Message_CreateBcToOrFromBroadcastMode, sitalBcControlWord_BUS_A, sitalBcControlWord_LOOPBACK_TEST_PERFORMED, sitalBcDataBlockSizeOption_DOUBLE, sitalBcMessageOption_DUAL_MESSAGE, sitalBcSetupOption_ASYNCHRONOUS_BOTH_PRIORITY_MODES, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_ASYNCHRONOUS_MESSAGE_ERROR, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalReturnCode_SUITABLE_ASYNCHRONOUS_MODE_UNDEFINED.

S16BIT _DECL
sitalBc_AsynchronousMessage_CreateBcToOrFromRtMode (S16BIT *swDevice*,
S16BIT *swMessageId*,
S16BIT *swDataBlockId*,
U16BIT *wRt*,
U16BIT *wMessageDirection*,
U16BIT *wModeCommand*,
U16BIT *wGapTime*,
U32BIT *dwOptions*,
U16BIT * *wapBuffer*
)

Create for given BC device an asynchronous message of type BC-to-RT-Mode or RT-to-BC-Mode, depending on given RT device-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX, respectively).

Note:

- An asynchronous message is asynchronously transmitted at low priority in result of calling function sitalBc_AsynchronousMessage_SendAtLowPriority.
- An asynchronous message is asynchronously transmitted at high priority in result of calling function sitalBc_AsynchronousMessage_SendAtHighPriority.
- Unlike a normal message, an asynchronous message may not be transmitted by user-defined commands; This is prohibited by function sitalBc_Command_Create.

Equivalent DDC definition: aceBCAsyncMsgCreateMode

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))

wRt	(in) Receiver/transmitter RT (sitalRtAddress_0-sitalRtAddress_31)
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX)
wModeCommand	(in) Mode command (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)
wapBuffer	(in) A pointer to an array of data words to be copied into the new data block

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_DataBlock_Create, sitalBc_Message_CreateBcToOrFromRtMode, sitalBcControlWord_BUS_A, sitalBcControlWord_LOOPBACK_TEST_PERFORMED, sitalBcDataBlockSizeOption_DOUBLE, sitalBcMessageOption_DUAL_MESSAGE, sitalBcSetupOption_ASYNCHRONOUS_BOTH_PRIORITY_MODES, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_ASYNCHRONOUS_MESSAGE_ERROR, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalReturnCode_SUITABLE_ASYNCHRONOUS_MODE_UNDEFINED.

```
S16BIT _DECL
sitalBc_AsynchronousMessage_CreateBcToRt ( S16BIT    swDevice,
                                           S16BIT    swMessageId,
                                           S16BIT    swDataBlockId,
                                           U16BIT    wReceiverRt,
                                           U16BIT    wReceiverRtSubaddress,
                                           U16BIT    wWordCount,
                                           U16BIT    wGapTime,
                                           U32BIT    dwOptions,
                                           U16BIT *  wapBuffer
                                         )
```

Create for given BC device:

- A double data block of given ID, and fill its allocated device memory with the contents of given buffer.
- An asynchronous message of type BC-to-RT that uses that data block.

Note:

- An asynchronous message is asynchronously transmitted at low priority in result of calling function sitalBc_AsynchronousMessage_SendAtLowPriority.
- An asynchronous message is asynchronously transmitted at high priority in result of calling function sitalBc_AsynchronousMessage_SendAtHighPriority.
- Unlike a normal message, an asynchronous message may not be transmitted by user-defined commands; This is prohibited by function sitalBc_Command_Create.

Equivalent DDC definition: aceBCAsyncMsgCreateBCtoRT

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wReceiverRt	(in) Receiver RT (sitalRtAddress_0-sitalRtAddress_31)
wReceiverRtSubaddress	(in) Receiver RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)

wWordCount	(in) Word count (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)
wapBuffer	(in) A pointer to an array of data words to be copied into the new data block

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_DataBlock_Create, sitalBc_Message_CreateBcToRt, sitalBcControlWord_BUS_A, sitalBcControlWord_LOOPBACK_TEST_PERFORMED, sitalBcDataBlockSizeOption_DOUBLE, sitalBcMessageOption_DUAL_MESSAGE, sitalBcSetupOption_ASYNCHRONOUS_BOTH_PRIORITY_MODES, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_ASYNCHRONOUS_MESSAGE_ERROR, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalReturnCode_SUITABLE_ASYNCHRONOUS_MODE_UNDEFINED.

```
S16BIT _DECL
sitalBc_AsynchronousMessage_CreateRtToBc ( S16BIT    swDevice,
                                           S16BIT    swMessageId,
                                           S16BIT    swDataBlockId,
                                           U16BIT    wTransmitterRt,
                                           U16BIT    wTransmitterRtSubaddress,
                                           U16BIT    wWordCount,
                                           U16BIT    wGapTime,
                                           U32BIT    dwOptions,
                                           U16BIT *  wapBuffer
                                         )
```

Create for given BC device an asynchronous message of type RT-to-BC.

Note:

- An asynchronous message is asynchronously transmitted at low priority in result of calling function sitalBc_AsynchronousMessage_SendAtLowPriority.
- An asynchronous message is asynchronously transmitted at high priority in result of calling function sitalBc_AsynchronousMessage_SendAtHighPriority.
- Unlike a normal message, an asynchronous message may not be transmitted by user-defined commands; This is prohibited by function sitalBc_Command_Create.

Equivalent DDC definition: aceBCAsyncMsgCreateRTtoBC

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wTransmitterRt	(in) Transmitter RT (sitalRtAddress_0-sitalRtAddress_31)
wTransmitterRtSubaddress	(in) Transmitter RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wWordCount	(in) Word count (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)

dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)
wapBuffer	(in) A pointer to an array of data words to be copied into the new data block

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_DataBlock_Create, sitalBc_Message_CreateRtToBc, sitalBcControlWord_BUS_A, sitalBcControlWord_LOOPBACK_TEST_PERFORMED, sitalBcDataBlockSizeOption_DOUBLE, sitalBcMessageOption_DUAL_MESSAGE, sitalBcSetupOption_ASYNCHRONOUS_BOTH_PRIORITY_MODES, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_ASYNCHRONOUS_MESSAGE_ERROR, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalReturnCode_SUITABLE_ASYNCHRONOUS_MODE_UNDEFINED.

S16BIT _DECL
sitalBc_AsynchronousMessage_CreateRtToBroadcast (S16BIT *swDevice*,
S16BIT *swMessageId*,
S16BIT *swDataBlockId*,
U16BIT *wReceiverRtSubaddress*,
U16BIT *wWordCount*,
U16BIT *wTransmitterRt*,
U16BIT *wTransmitterRtSubaddress*,
U16BIT *wGapTime*,
U32BIT *dwOptions*,
U16BIT * *wapBuffer*
)

Create for given BC device an asynchronous message of type RT-to-Broadcast.

Note:

- An asynchronous message is asynchronously transmitted at low priority in result of calling function sitalBc_AsynchronousMessage_SendAtLowPriority.
- An asynchronous message is asynchronously transmitted at high priority in result of calling function sitalBc_AsynchronousMessage_SendAtHighPriority.
- Unlike a normal message, an asynchronous message may not be transmitted by user-defined commands; This is prohibited by function sitalBc_Command_Create.

Equivalent DDC definition: aceBCAsyncMsgCreateBcstRTtoRT

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))

swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wReceiverRtSubaddress	(in) Receiver RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wWordCount	(in) Word count (1-32)
wTransmitterRt	(in) Transmitter RT (sitalRtAddress_0-sitalRtAddress_31)
wTransmitterRtSubaddress	(in) Transmitter RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)
wapBuffer	(in) A pointer to an array of data words to be copied into the new data block

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_DataBlock_Create, sitalBc_Message_CreateRtToBroadcast, sitalBcControlWord_BUS_A, sitalBcControlWord_LOOPBACK_TEST_PERFORMED, sitalBcDataBlockSizeOption_DOUBLE, sitalBcMessageOption_DUAL_MESSAGE, sitalBcSetupOption_ASYNCHRONOUS_BOTH_PRIORITY_MODES, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_ASYNCHRONOUS_MESSAGE_ERROR, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalReturnCode_SUITABLE_ASYNCHRONOUS_MODE_UNDEFINED.

```
S16BIT _DECL
sitalBc_AsynchronousMessage_CreateRtToRt ( S16BIT   swDevice,
                                           S16BIT   swMessageId,
                                           S16BIT   swDataBlockId,
                                           U16BIT   wReceiverRt,
                                           U16BIT   wReceiverRtSubaddress,
                                           U16BIT   wWordCount,
                                           U16BIT   wTransmitterRt,
                                           U16BIT   wTransmitterRtSubaddress,
                                           U16BIT   wGapTime,
```

```
    U32BIT   dwOptions,
    U16BIT
    *        wapBuffer
)
```

Create for given BC device an asynchronous message of type RT-to-RT.

Note:

- An asynchronous message is asynchronously transmitted at low priority in result of calling function sitalBc_AsynchronousMessage_SendAtLowPriority.
- An asynchronous message is asynchronously transmitted at high priority in result of calling function sitalBc_AsynchronousMessage_SendAtHighPriority.
- Unlike a normal message, an asynchronous message may not be transmitted by user-defined commands; This is prohibited by function sitalBc_Command_Create.

Equivalent DDC definition: aceBCAsyncMsgCreateRTtoRT

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wReceiverRt	(in) Receiver RT (sitalRtAddress_0-sitalRtAddress_31)
wReceiverRtSubaddress	(in) Receiver RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wWordCount	(in) Word count (1-32)
wTransmitterRt	(in) Transmitter RT (sitalRtAddress_0-sitalRtAddress_31)
wTransmitterRtSubaddress	(in) Transmitter RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)
wapBuffer	(in) A pointer to an array of data words to be copied into the new data block

Returns:

sitalReturnCode_SUCCESS Function successfully completed

Negative sitalReturnCode_* Error condition or function failed

References sitalBc_DataBlock_Create, sitalBc_Message_CreateRtToRt, sitalBcControlWord_BUS_A, sitalBcControlWord_LOOPBACK_TEST_PERFORMED, sitalBcDataBlockSizeOption_DOUBLE, sitalBcMessageOption_DUAL_MESSAGE, sitalBcSetupOption_ASYNCHRONOUS_BOTH_PRIORITY_MODES, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_ASYNCHRONOUS_MESSAGE_ERROR, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalReturnCode_SUITABLE_ASYNCHRONOUS_MODE_UNDEFINED.

S16BIT _DECL
sitalBc_AsynchronousMessage_SendAtHighPriority (S16BIT *swDevice*,
U16BIT *wMessageId*,
U16BIT *wTimeFactor*
)

Fix for given BC device the subroutine that asynchronously transmits a single desired message at high priority, so that it will transmit given message. Then raise the proper general purpose flag, GPF-6, so that this subroutine will be actually called along the currently ongoing frame run. Finally, wait for given message to be transmitted.

Note:

- Given message must be an asynchronous message.
- Given message must be a dual message in which the second message's message block command field is 0xFFFF, which is an illegal message; This is the kind of messages automatically created by the sitalBc_AsynchronousMessage_Create* functions. This kind of messages, once successfully transmitted using a XQF opcode that's conditioned with "good message" (which is the command used by the high priority asynchronous message Tx subroutine), makes the device flip the message address to this dual message's second message, where the 0xFFFFU makes the device avoid further Tx trials and message flips, so no time will be spent anymore with this message.
- Once an asynchronous message is defined, it may be transmitted at high priority again and again as many times as the user requires, but it won't be transmitted more than once at low priority. Moreover, if an asynchronous message has already been transmitted at high priority, it won't be retransmitted at low priority even once. In order to force its retransmission in such a case, the message must be deleted and recreated.
- This function does not really require the time factor parameter, which isn't removed only in order to stay compatible with DDC.
- Calling the subroutines that perform the high & low priority asynchronous message Tx (including their initial stubs) is conditioned by GPF-6 & GPF-7, respectively, being on; User applications should restrict themselves to never use GPF-6 and GPF-7!

Equivalent DDC definition: aceBCSendAsyncMsgHP

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
wTimeFactor	(in) A time factor (Unused)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBcConditionCode_GPF6, sitalBcGpfEffect_CLEAR, sitalBcGpfEffect_SET, sitalBcSetupOption_ASYNCHRONOUS_HIGH_PRIORITY_MODE, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalProcess_Log_PrintLine(), sitalRegisterAddress_BC_INSTRUCTION_LIST_POINTER, sitalReturnCode_ASYNCHRONOUS_MESSAGE_ERROR, sitalReturnCode_ASYNCHRONOUS_SUBROUTINE_BUSY, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalReturnCode_SUITABLE_ASYNCHRONOUS_MODE_UNDEFINED.

S16BIT _DECL
sitalBc_AsynchronousMessage_SendAtLowPriority (S16BIT *swDevice*,
U16BIT * *wpRemainingMessageCount*,
U16BIT *wTimeFactor*
)

Fix for given BC device the subroutine that asynchronously transmits messages at low priority, so that it will transmit all the currently defined asynchronous messages. Then raise the proper general purpose flag, GPF-7, so that this subroutine will be actually called along the currently ongoing frame run.

Note:

- For an asynchronous message to be transmitted by this function it must be a dual message in which the second message's message block command field is 0xFFFFU; This is the kind of messages automatically created by the sitalBc_AsynchronousMessage_Create* functions. This kind of messages, once successfully transmitted using a XQF opcode that's conditioned with "good message" (which is the command used by the high priority asynchronous message Tx subroutine), makes the device flip the message address to this dual message's second message, where the 0xFFFFU makes the device avoid further Tx trials and message flips.
- Once an asynchronous message is defined, it may be transmitted at high priority again and again as many times as the user requires, but it won't be transmitted more than once at low priority. Moreover, if an asynchronous message has already been transmitted at high priority, it won't be retransmitted at low priority even once. In order to force its retransmission in such a case, the message must be deleted and recreated.
- Calling the subroutines that perform the high & low priority asynchronous message Tx (including their initial stubs) is conditioned by GPF-6 & GPF-7, respectively, being on. In order to maintain this mechanism intact, user applications should never use GPF-6 and GPF-7.
- The count of asynchronous messages that still weren't assigned for Tx, being set for the variable pointed by wpRemainingMessageCount, is actually always zero. Accordingly, the returned count of asynchronous messages assigned for Tx in the new low priority asynchronous message Tx subroutine is actually the total number of currently defined asynchronous messages that still weren't asynchronously transmitted at either high or low priority. In contrary to the corresponding library of DDC, all those asynchronous message that weren't yet transmitted are set for Tx by the low priority asynchronous message Tx microcode subroutine that's built by this function, nevermind how much free passive time is available at the end of the currently executed minor frames. This policy is possible due to the fact that this library leaves the decision of whether to transmit more low priority asynchronous messages or not to the device itself, which will continue Tx only as long as

there's enough time for more messages, and only with messages small enough to fit the time left.

- This function does not really require the time factor parameter, which isn't removed only in order to stay compatible with DDC.
- This function does not wait for the low priority asynchronous message Tx to complete; Instead, the user may recall this function again and again, possibly with a small delay in between the calls, until informed that there are no more asynchronous messages to transmit.
- The low priority asynchronous message Tx subroutine is made of a series of command quartets, one for each of the asynchronous messages that weren't transmitted yet, where each of these quartets contains the following commands:
    - A command to compare the frame timer to 2 ms.
    - A jump command to-the-end-of-this-subroutine if-less-than
    - An execute-and-flip command of-the-next-asynchronous-message if-good-message
    - A command to call the high priority asynchronous message Tx subroutine conditioned by GPF-6 being raised. Finally, a return command is added, in order to return control to the currently executing minor frame.
- The transmission of low priority asynchronous messages is supported by this library only while running a major frame that forces a common frame time over all its minor frames. This enables it to concentrate the transmission commands for all the asynchronous messages in a single common routine and this way save device memory space. What's more, this policy leaves the decision of which asynchronous messages to transmit at the end of each minor frame for the device, which may then dynamically and accurately decide about it, taking in account the actual time it took to transmit the messages of each frame in the current occasion.

Equivalent DDC definition: aceBCSendAsyncMsgLP

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wpRemainingMessageCount	(out) A pointer to a variable in which the count of asynchronous messages that still weren't assigned for Tx is returned
wTimeFactor	(in) A time factor (Unused)

Returns:

Positive integer The number of asynchronous messages assigned for Tx in the new low priority asynchronous message Tx subroutine
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_Command_GetWord(), sitalBcConditionCode_GPF7, sitalBcCounter_MESSAGES, sitalBcGpfEffect_CLEAR, sitalBcGpfEffect_SET, sitalBcMaximum_ASYNCHRONOUS_MESSAGES, sitalBcSetupOption_ASYNCHRONOUS_LOW_PRIORITY_MODE, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceMemorySection_Registers, sitalDeviceState_RUN, sitalMode_BC, sitalOpcode_CALL_SUBROUTINE, sitalOpcode_COMPARE_TO_FRAME_TIMER, sitalOpcode_EXECUTE_AND_FLIP, sitalOpcode_JUMP, sitalOpcode_RETURN_FROM_SUBROUTINE, sitalOpcodeCondition_ALWAYS, sitalOpcodeCondition_GOOD_MESSAGE, sitalOpcodeCondition_GP6_1, sitalOpcodeCondition_GREATER_THAN, sitalRegisterAddress_BC_INSTRUCTION_LIST_POINTER, sitalReturnCode_ALLOCATION_FAIL, sitalReturnCode_ASYNCHRONOUS_LIST_IS_EMPTY, sitalReturnCode_ASYNCHRONOUS_SUBROUTINE_BUSY, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_NO_FRAME_TIME_SET_FOR_MAJOR_FRAME, sitalReturnCode_SUCCESS, and sitalReturnCode_SUITABLE_ASYNCHRONOUS_MODE_UNDEFINED.

S16BIT _DECL
sitalBc_Command_Create (S16BIT *swDevice*,
S16BIT *swCommandId*,
U16BIT *wOpcode*,
U16BIT *wCondition*,
U32BIT *dwParameter1*,
U32BIT *dwParameter2*,
U32BIT *dwReserved*
)

Create for given BC device a new command using given command descriptors. The new command will be actually allocated device memory space and written in it only at the right moments along the course of the life of its frame.

Note:

- Given opcode may be either:
  - Hardware (=regular/standard) opcodes.
  - Software (=special/non-standard) opcodes: Opcodes that designate [frequently used] series of hardware opcodes. Here's the list of the currently defined software opcodes and their translation into hardware opcodes:
    - sitalOpcode_TIME_CONDITIONED_MESSAGE_TX (condition, parameter): 00: ... (The preceding part of the frame) 01: JMP if the given condition is not fulfilled to address 07 02: CFT with watchdog time plus message time 03: JMP if greater than to address 07 04: XEQ given message 05: FLG to clear the relevant GPF, so that given condition won't fulfill anymore (if depends on some GPF) 06: CAL the high priority asynchronous messages Tx subroutine (only if high priority asynchronous messages are supported) 07: ... (The rest of the frame)
- Here are parameter and other important specifications for opcodes for which any parameters are defined:
  - Message-transmitter commands (execute message, execute-and-flip, and time conditioned message Tx):
    - Parameter #1: The ID of the normal (i.e., not asynchronous) message to transmit.
  - Jump:
    - Parameter #1: The backward/forward offset (in commands) to jump in two's-complement representation (e.g., 0xFFFFU means jump one command backward).
  - Call subroutine:

        - Parameter #1: The ID of the minor (i.e., not major) frame to execute at this point.
    - Return from subroutine:
        - WARNING: This command should not be used within a major frame!
    - Interrupt request:
        - Parameter #1: Its 4 LSbits contain the index of the IRQ to issue (e.g., a binary 0100, or 0x4, means IRQ #2).
    - Delay:
        - Parameter #1: Time (in us) to delay.
    - Compare to frame timer:
        - Parameter #1: Time stamp (in 100us).
    - Compare to message timer:
        - Parameter #1: Time stamp (in us).
    - Update GPF:
        - Parameter #1:
            - If bit #i of the 8 LSbits and none of the 8 MSbits is on, set GPF #i.
            - If bit #i of the 8 MSbits and none of the 8 LSbits is on, clear GPF #i.
            - If bit #i of both the 8 LSbits and of the 8 MSbits is on, toggle GPF #i.
    - Load time tag counter:
        - Parameter #1: Value to load (the resolution is determined by configuration register #2).
    - Load frame time:
        - Parameter #1: Time stamp (in 100us).
    - Push value to the GPQ:
        - WARNING: This library reports frame run events by pushing pairs of suitable word values into the GPQ, as follows:
            - Header word: A special values that designates that exact reported event (sitalGeneralPurposeQueueFlag_*).
            - Data word: Some relevant data that further describes that reported event. The library later reads these GPQ entries in order to see what happened. Therefore, in order not to disturn this apparatus, the user must: - In each such pair, tha first PUSH must write into the GPQ a value that differs from those above specified GPQ headers that are used by this library.
        - Parameter #1: A value to push.
    - Indirect push of value to the GPQ:
        - Parameter #1: A device memory address to take the value to push from.
    - Focus on variable:
        - Parameter #1: The 4 LSbits contain the number of the variable to focus on.
        - WARNING: Variables #8 and #9 are used by this library, and therefore should not be used!

    - Push variable value to the GPQ:
        - Parameter #1: The 4 LSbits contain the number of the desired variable.
    - Add to the now focused variable:
        - Parameter #1: A signed value to add, either negative or positive, in two's-complement representation.
    - Load the now focused variable:
        - Parameter #1: A value to load.
    - Compare the now focused variable:
        - Parameter #1: A value to compare.
- In cases where this function does not reject commands with illegal parameter values, they'll be eventually rejected when their container frame comes to execution.

Equivalent DDC definition: aceBCOpCodeCreate

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swCommandId	(in) A unique ID designating a command (0-(sitalBcCounter_COMMANDS-1))
wOpcode	(in) The opcode operated by the new command (sitalOpcode_*)
wCondition	(in) The condition that must be fulfilled in order for the command to be executed (sitalOpcodeCondition_*)
dwParameter1	(in) A parameter that's possibly required by some opcodes (See the opcodes description table under 'Instructions' in the eBC's user manual)
dwParameter2	(in) Another parameter that's possibly required by some opcodes (See the opcodes description table under 'Instructions' in the eBC's user manual)
dwReserved	(in) Reserved for future use

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBcCounter_COMMANDS, sitalBcCounter_FRAMES, sitalDeviceState_READY, sitalMode_BC, sitalOpcode_CALL_SUBROUTINE, sitalOpcode_EXECUTE_AND_FLIP, sitalOpcode_EXECUTE_MESSAGE, sitalOpcode_TIME_CONDITIONED_MESSAGE_TX, sitalReturnCode_ASYNCHRONOUS_MESSAGE_ERROR, sitalReturnCode_BC_OBJECT_ALREADY_EXISTS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL sitalBc_Command_Delete ( S16BIT  swDevice,
                                      S16BIT  swCommandId
                                    )
```

Delete given command which has been previously created for given BC device.

Note:

- A command may be principally deleted along frame run in order to be compatible with DDC's library; Yet, delete isn't supported if this command is used by any existing frame, even though as such an operation won't for itself directly influence the ongoing frame run, as this function only deletes the host memory representation of given command, and device memory/operation isn't affected at all by it; Read about the reason for this in the documentation for function sitalBc_Frame_Delete.

Equivalent DDC definition: aceBCOpCodeDelete

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swCommandId	(in) A unique ID designating a command (0-(sitalBcCounter_COMMANDS-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBcCounter_COMMANDS, sitalBcCounter_FRAMES, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_NOT_SUPPORTED, sitalReturnCode_SUCCESS, and sitalReturnCode_UNDEFINED_NODE.

S16BIT _DECL
sitalBc_Command_DistortWord (U16BIT *wHardwareOpcode*,
U16BIT *wDistortionManner*,
U16BIT *wDistortionParameter*
)

Configure function sitalBc_Command_GetWord of the stld1553 library to insert from now on given distort to its generated commands.

Note:

- This function is only supplied for test purposes, and isn't intended for regular use! Once called, it may cause the creation of faulty frames.
- Given distortion settings will affect any BC device initialized by the calling app.

Parameters:

wHardwareOpcode	(in) An hardware opcode whose commands should be distorted from now on (sitalOpcode_*)
wDistortionManner	(in) A requested manner of command distortion (sitalOpcodeDistortion_*)
wDistortionParameter	(in) An optionally required parameter as specified in the definition of given distortion manner

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalOpcodeDistortion_FlipBit, sitalOpcodeDistortion_None, sitalReturnCode_INVALID_PARAMETER, and sitalReturnCode_SUCCESS.

S16BIT _DECL sitalBc_Command_GetWord (U16BIT *wHardwareOpcode*,
U16BIT *wCondition*,
U16BIT * *wpCommandOpcode*
)

Generate a command word using given opcode and condition code. Odd parity is used for the generation of the parity bit (MSb) of the command opcode word.

Note:

- Given hardware opcode may be either a regular opcode or one of Sital's unique hardware opcodes. The latter ones are numbered by this library with unique artificial numbers, but must be converted to their true values when recorded in device memory. As result, there are some opcode numbers that represent a pair of opcodes, that is both a regular and a Sital unique opcode. The way for the device to know whether an opcode is a regular or a Sital unique one is to examine the mandatory bits.

Parameters:

wHardwareOpcode	(in) An hardware opcode (sitalOpcode_*)
wCondition	(in) A condition code (sitalOpcodeCondition_*)
wpCommandOpcode	(out) A pointer to a variable within which the generated command opcode is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalOpcode_FOCUS_ON_VARIABLE, sitalOpcode_PUSH_VARIABLE, sitalOpcodeCondition_ALWAYS, sitalOpcodeCondition_NEVER, sitalOpcodeDistortion_FlipBit, sitalOpcodeDistortion_None, sitalOpcodeDistortion_SetCommand, sitalReturnCode_INVALID_PARAMETER, and sitalReturnCode_SUCCESS.

Referenced by sitalBc_AsynchronousMessage_SendAtLowPriority(), sitalBc_DataBlock_Read(), and sitalBc_Start().

S16BIT _DECL
sitalBc_ConfigureMessageGapTimeFieldPolicy (S16BIT *swDevice*,
U16BIT *bIsMessageGapTimeFieldEnabled*
)

Configure given BC device to enable/disable the message gap time field for all messages.

Equivalent DDC definition: aceBCMsgGapTimerEnable

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
bIsMessageGapTimeFieldEnabled	(in) A flag that says whether the message gap time field should be enabled for all messages (or, otherwise, disabled)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister1_BC_MESSAGE_GAP_TIME_FIELD_ENABLED, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalRegisterAddress_CONFIGURATION_1, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalBc_DataBlock_Create ( S16BIT    swDevice,
                           S16BIT    swDataBlockId,
                           U16BIT    wDataBlockSize,
                           U16BIT *  wapBuffer,
                           U16BIT    wBufferSize
                         )
```

Create for given BC device a data block of given ID and size, and fill its allocated device memory with the contents of given buffer.

Note:

- An actual data block may contain up to 32 data words, but this function automatically allocates 32 words of device memory for all data blocks.
- Its address/offset (in word) in device memory must be a number whose 5 LSBits are zeros.
- With data blocks that are used with mode code messages with data, the single data word is always read from or written into the first data word of the data block.

Equivalent DDC definition: aceBCDataBlkCreate

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wDataBlockSize	(in) The size (in words) of the new data block (1-32 or sitalBcDataBlockSizeOption_DOUBLE)
wapBuffer	(in) A pointer to an array of data words to be copied into the new data block, or NULL if this buffer isn't required
wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (0-wDataBlockSize)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_DataBlock_Write, sitalBcCounter_DATA_BLOCKS, sitalBcDataBlockSizeOption_DOUBLE, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_ALLOCATION_FAIL, sitalReturnCode_BC_OBJECT_ALREADY_EXISTS,

sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalBc_DataBlock_Delete (S16BIT *swDevice*,
S16BIT *swDataBlockId*
)

Delete given data block which has been previously created for given BC device, and free the segment of device memory that has been allocated for it.

Note:

- A data block may be principally deleted along frame run in order to be compatible with DDC's library; Yet, delete isn't supported if this data block is used by any existing message, so that a situation in which a transmitted message tries to use an already deleted data block is prohibited.

Equivalent DDC definition: aceBCDataBlkDelete

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBcCounter_DATA_BLOCKS, sitalBcCounter_MESSAGES, sitalBcMessageOption_DUAL_MESSAGE, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_NOT_SUPPORTED, sitalReturnCode_SUCCESS, and sitalReturnCode_UNDEFINED_NODE.

S16BIT _DECL sitalBc_DataBlock_Read (S16BIT *swDevice*,
S16BIT *swDataBlockId*,
U16BIT * *wapBuffer*,
U16BIT *wBufferSize*,
U16BIT *wOffset*
)

Read the contents of given data block of given BC device starting at given offset into given buffer. For this purpose consider the data block as a cyclic buffer, i.e., continue the copy operation from its start if/when reaching its end.

Note:

- With IEEE-1553 a message data block is up to 32 words long, where the actual number of words depends on the definition of the message. Now, since this function knows nothing about the message that uses or will use given data blobk, it lets the caller read up to 32 words from it. Taking this approach, the user is responsible to write only words that will be used by the message that eventually points given data block.
- In case of a dual data block and a dual message that's transmitted using a XQF command:
  - The first data block is used by the first message.
  - The second data block is used by the second message. After a XQF command is executed with either its first/second message, then if the condition is fulfilled, the device turns to the other message. When the device is about to execute one of such two messages, this function first prevents it from flipping to the other message, then it reads data from the data block of the other message (which is the one previously updated by the device), and then reenables the device to perform flips.
- With data blocks that are used with mode code messages with data, the single data word is always read from or written into the first data word of the data block.

Equivalent DDC definition: aceBCDataBlkRead

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wapBuffer	(in) A pointer to an array of data words to be copied into the new data block
wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (0-<size-of-given-data-block>)
wOffset	(in) The offset (in words) from the beginning of the data block in device memory where cyclic copy begins (0-(<size-of-given-data-block>-1))

Returns:

Positive integer The number of words that were read into given buffer
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_Command_GetWord(), sitalBcCounter_DATA_BLOCKS, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram,

sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalOpcode_EXECUTE_AND_FLIP, sitalOpcodeCondition_GOOD_MESSAGE, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalReturnCode_UNDEFINED_NODE.

```
S16BIT _DECL sitalBc_DataBlock_Write ( S16BIT    swDevice,
                                       S16BIT    swDataBlockId,
                                       U16BIT *  wapBuffer,
                                       U16BIT    wBufferSize,
                                       U16BIT    wOffset
                                     )
```

Write the contents of given buffer in given data block of given BC device at given offset. For this purpose consider the data block as a cyclic buffer, i.e., continue the copy operation from its start if/when reaching its end.

Note:

- With IEEE-1553 a message data block is up to 32 words long, where the actual number of words depends on the definition of the message. Now, since this function knows nothing about the message that uses or will use given data blobk, it lets the caller write up to 32 words into it. Taking this approach, the user is responsible to write only words that will be used by the message that eventually points given data block.
- In case of a dual data block and a dual message that's transmitted using a XQF command:
  - The first data block is used by the first message.
  - The second data block is used by the second message. After a XQF command is executed with either its first/second message, then if the condition is fulfilled, the device turns to the other message. When the device is about to execute one of such two messages, this function writes to the data block of the other message in prepare for the next turn, and then performs a flip.
- With data blocks that are used with mode code messages with data, the single data word is always read from or written into the first data word of the data block.

Equivalent DDC definition: aceBCDataBlkWrite

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))

wapBuffer	(in) A pointer to an array of data words to be copied into the new data block
wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (0-<size-of-given-data-block>)
wOffset	(in) The offset (in words) from the beginning of the data block in device memory where cyclic copy begins (0-(<size-of-given-data-block>-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBcCounter_DATA_BLOCKS, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalReturnCode_UNDEFINED_NODE.

```
S16BIT _DECL
sitalBc_Frame_Create ( S16BIT   swDevice,
                       S16BIT   swFrameId,
                       U16BIT   wFrameType,
                       S16BIT * swapCommandIds,
                       U16BIT   wCommandCount,
                       U16BIT   wFrameTime,
                       U16BIT   wFlags
                     )
```

Create for given BC device a new frame and calculate the overall number of included commands.

Note:

- Recursion isn't allowed: A frame may not include a call to a subroutine that execute itself either directly or indirectly (i.e., via other frames); Yet, for the sake of simplicity and in order to let the user application more freedom in its frame creation sequence, this function only rejects direct recursions: indirect recursions are identified and rejected only later on, when the user tries to run a frame.
- A frame that its later on explicit (=non-recursive) execution is intended, must be classified as major, not minor.
- Given frame time is interpreted as follows:
    - If zero, it is considered as unspecified.
    - Otherwise, if that's a:
        - Major frame: Once this major frame is run, given time will be forced for all nested minor frames.
        - Minor frame: Once this minor frame is executed from within some major frame, given time will take effect with this minor frame if and only if the containing major frame does not force a frame time of its own.
- This function doesn't really allocate device memory for the new frame: That will be done only once the user app requests to run this frame. When a major frame is run using function sitalBc_Start, the major frame is actually built in device memory, together with all its nested minor frames, and for this purpose special framing commands are then added (see full information in the documentation for function sitalBc_Start).

Equivalent DDC definition: aceBCFrameCreate

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))

swFrameId	(in) A unique ID designating a frame (0-(sitalBcCounter_FRAMES-1))
swFrameType	(in) The type of the new frame (sitalBcFrameType_*)
swapCommandIds	(in) A pointer to an array of opcode IDs that make up the new frame (1-(sitalBcCounter_COMMANDS-<required-number-of-framing-commands>))
wCommandCount	(in) The number of command IDs recorded in the array that is pointed by swpCommandIds (1-sitalBcCounter_FRAME_COMMANDS)
wFrameTime	(in) The time (in units of 100 us) that the frame will take to complete
wFlags	(in) Frame configuration options (sitalBcFrameFlag_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBcCounter_COMMANDS, sitalBcCounter_FRAME_COMMANDS, sitalBcCounter_FRAMES, sitalBcFrameType_MAJOR, sitalBcSetupOption_ASYNCHRONOUS_HIGH_PRIORITY_MODE, sitalDeviceState_READY, sitalMode_BC, sitalOpcode_CALL_SUBROUTINE, sitalOpcode_EXECUTE_AND_FLIP, sitalOpcode_EXECUTE_MESSAGE, sitalOpcode_TIME_CONDITIONED_MESSAGE_TX, sitalReturnCode_BC_OBJECT_ALREADY_EXISTS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_NOT_SUPPORTED, sitalReturnCode_SUCCESS, and sitalReturnCode_UNDEFINED_COMMAND.

S16BIT _DECL **sitalBc_Frame_Delete** (S16BIT *swDevice*,
S16BIT *swFrameId*
)

Delete given frame which has been previously created for given BC device.

Note:

- A frame may not be deleted along frame run, though as such an operation won't for itself directly influence the ongoing frame run, as this function only deletes the host memory representation of given frame, and device memory/operation isn't affected at all by it; The reason for this restriction is that if the user is allowed to delete frames along frame run, he/she may delete the commands that where included in the deleted frames, and in turn

delete messages transmitted by these commands, and data blocks used by these messages, what may disrupt the execution of the currently running frame.

- A frame may not be deleted if it is used by any existing frame.

Equivalent DDC definition: aceBCFrameDelete

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

swFrameId (in) A unique ID designating a frame (0-(sitalBcCounter_FRAMES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed

Negative sitalReturnCode_* Error condition or function failed

References sitalBcCounter_FRAMES, sitalDeviceState_READY, sitalMode_BC, sitalOpcode_CALL_SUBROUTINE, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_NOT_SUPPORTED, sitalReturnCode_SUCCESS, and sitalReturnCode_UNDEFINED_NODE.

S16BIT _DECL\
sitalBc_GetActivationState (S16BIT *swDevice*,
U16BIT * *wpCurrentState*
)

Get the activation state, idle or busy, of given BC device.

Note:

- A BC device becomes busy when it starts running a frame, and automatically returns to be idle when all required frame iterations are finished (i.e., halt command reached).

Equivalent DDC definition: aceBCGetStatus

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wpCurrentState	(out) A pointer to a variable within which current state, idle (sitalBcActivationState_IDLE) or busy (sitalBcActivationState_BUSY), is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBcActivationState_BUSY, sitalBcActivationState_IDLE, sitalConfigurationRegister1_BC_BUSY, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Registers, sitalMode_BC, sitalRegisterAddress_CONFIGURATION_1, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalBc_GetConditionState (S16BIT *swDevice*,
U16BIT *wCondition*,
U16BIT * *wpCurrentState*
)

Read the BC Condition Code register of given BC device and obtain the current state of given condition.

Note:

- The current state of given condition is the value currently stored in its respective bit[s] in the BC condition code register.

Equivalent DDC definition: aceBCGetConditionCode

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wCondition	(in) Condition code (sitalBcConditionCode_*)
wpCurrentState	(out) A pointer to a variable within which current state, off (0) or on (1), is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister1_BC_BUSY, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Registers, sitalDeviceState_RUN, sitalMode_BC, sitalRegisterAddress_BC_CONDITION_CODE, sitalRegisterAddress_CONFIGURATION_1, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalBc_Gpf_SetState ( S16BIT  swDevice,
                       U16BIT  wGpfNumber,
                       U16BIT  wEffect
                     )
```

Set given effect with the state of given GFP condition of given BC device.

Equivalent DDC definition: aceBCSetGPFState

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wGpfNumber	(in) General-purpose-flag-related condition code (sitalBcGpf_GPF*)
wEffect	(in) Requested effect (sitalBcGpfEffect_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, and sitalReturnCode_INVALID_STATE.

S16BIT _DECL
sitalBc_Gpq_GetEntriesCount (S16BIT *swDevice*)

Return the current number of available entries in the GPQ of given BC device.

Note:

- This function only considers GPQ entries that were filled by BC commands that have been explicitly inserted to the running frame by the user. In other words, those GPQ entries that are regularly filled by BC commands that have been inserted to the running frame by this library (i.e., framing commands, see the documentation for function sitalBc_Start) aren't taken in account by this function.

Equivalent DDC definition: aceBCGPQGetCount

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

Negative sitalReturnCode_* Error condition or function failed
Non-negative integer Count of available GPQ entries

References sitalBcCounter_GPQ_ENTRIES, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, and sitalReturnCode_INVALID_STATE.

S16BIT _DECL
sitalBc_Gpq_GetOperationalStatistics (S16BIT *swDevice*,
sitalGpqOperationalStatisticsStructure * *gosspGpqOperationalStatistics*,
U16BIT *bIsResetOfHighestRecordedPercentageRequired*
)

Return performance information about the general purpose queue of given BC device.

Equivalent DDC definition: aceBCGetGPQMetric

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
gosspGpqOperationalStatistics	(out) A pointer to the general purpose queue operational statistics structure into which the required operational statistics are written
bIsResetOfHighestRecordedPercentageRequired	(in) A flag that says whether the record of the highest percentage reached by now should be reset

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_OPERATIONAL_STATISTICS_NOT_ENABLED, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalBc_Gpq_HandleNewEntries (S16BIT *swDevice*)

Handle the new entries of the GPQ of given BC device. The way each GPQ entry is handled depends on its contents, and may include making some internal updates, calling the user application's ISR, and recording the latest minor frame that was transmitted by given BC device in its host buffer, if one is available.

Note:

- As can be seen in the documentation for function sitalBc_Start, the scheme used by this library when programming a target BC device with minor frames includes the issue of the following interrupts:
  - IRQ-3: At end of the active part of each minor frame, if only so configured for this minor frame (see the possible values for parameter wFlags of function sitalBc_Frame_Create).
  - IRQ-2: At successful end of the passive part (that follows the active part) of each minor frame, if only high-priority asynchronous messages are supported and host buffer has been assigned with the target device. Whenever any of these interrupts occurs, as well as in case of time tag rollover interrupts, this function gets automatically called.
- Though not recommended, a user application may also call this function, an resort option with systems where interrupts aren't supported.

Equivalent DDC definition: aceBCFrmToHBuf

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBcCounter_FRAMES, sitalBcCounter_GPQ_ENTRIES, sitalBcFrameType_MINOR, sitalDeviceState_READY, sitalDeviceState_RUN, sitalGeneralPurposeQueueFlag_END_OF_MAJOR_FRAME, sitalGeneralPurposeQueueFlag_END_OF_MINOR_FRAME, sitalGeneralPurposeQueueFlag_WATCHDOG_TIMEOUT, sitalMode_BC, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalReturnCode_SYNCHRONIZATION_FAIL.

S16BIT _DECL **sitalBc_Gpq_Read** (S16BIT *swDevice*,
sitalGpqEntryStructure * *gespGpqEntry*
)

Read the next unread entry from the GPQ of given BC device.

Note:

- This function only returns GPQ entries that were filled by BC commands that have been explicitly inserted to the running frame by the user. In other words, those GPQ entries that are regularly filled by BC commands that have been inserted to the running frame by this library (i.e., framing commands, see the documentation for function sitalBc_Start) aren't returned by this function.
- Yet, this function also reads GPQ entries that were filled by framing commands, internally stores them, and removes them from the GPQ.

Equivalent DDC definition: aceBCGPQRead

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
gespGpqEntry	(out) A pointer to the GPQ entry structure to fill with the now-read GPQ entry

Returns:

sitalReturnCode_SUCCESS Function successfully completed
sitalReturnCode_ReadGpqEntry_NO_NEW_ONES No entries were read
sitalReturnCode_ReadGpqEntry_READ_NEW_ENTRY One entry was read
sitalReturnCode_ReadGpqEntry_READ_NEW_ENTRY_AND_DETECTED_OVERRUN One entry was read, and one other or more were lost
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, and sitalReturnCode_INVALID_STATE.

S16BIT _DECL
sitalBc_HostBuffer_Free (S16BIT *swDevice*)

Free given bc device's host buffer.

Equivalent DDC definition: aceBCUninstallHBuf

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalDeviceState_READY, sitalInterruptRegister1_TIME_TAG_ROLLOVER, sitalMode_BC, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalBc_HostBuffer_GetOperationalStatistics (
    S16BIT                                  swDevice,
    sitalHostBufferOperationalStatisticsStructure *  hbosspHostBufferOperationalStatistics,
    U16BIT                                  bIsResetOfHighestRecordedPercentageRequired
)
```

Return performance information about the host buffer of given BC device.

Equivalent DDC definition: aceBCGetHBufMetric

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
hbosspHostBufferOperationalStatistics	(out) A pointer to the host buffer operational statistics structure into which the required operational statistics are written
bIsResetOfHighestRecordedPercentageRequired	(in) A flag that says whether the record of the highest percentage reached by now should be reset

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_OPERATIONAL_STATISTICS_NOT_ENABLED, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalBc_HostBuffer_Initialize (S16BIT *swDevice*,
U32BIT *dwHostBufferSize*
)

Initialize (or re-initialize) given BC device's host buffer.

Equivalent DDC definition: aceBCInstallHBuf

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dwHostBufferSize	(in) The desired size (in words) of the host buffer (sitalMinimum_SIZE_OF_HOST_BUFFER-sitalMaximum_SIZE_OF_HOST_BUFFER)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalBc_HostBuffer_Free, sitalBcMaximum_MESSAGE_SIZE, sitalDeviceState_READY, sitalInterruptRegister1_TIME_TAG_ROLLOVER, sitalMaximum_SIZE_OF_HOST_BUFFER, sitalMinimum_SIZE_OF_HOST_BUFFER, sitalMode_BC, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_HOST_BUFFER_SIZE, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalBc_HostBuffer_Message_GetCount (S16BIT *swDevice*)

Get the number of messages that are currently available in the host buffer that is assigned with given bc device.

Equivalent DDC definition: aceBCGetHBufMsgCount

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

Non-negative integer The number of messages that are currently available in the host buffer that is assigned with given device
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, and sitalReturnCode_INVALID_STATE.

S16BIT _DECL
sitalBc_HostBuffer_Message_GetDecoded (S16BIT *swDevice*,
sitalDecodedMessageStructure * *dmspDecodedMessage*,
U32BIT * *dwpMessageCount*,
U32BIT * *dwpLostMessageCount*,
U16BIT *wMessageLocationAndRemoval*
)

Read from the host buffer of given BC device the message at given location, and decode it into given structure. Remove from the host buffer the message that was read.

Equivalent DDC definition: aceBCGetHBufMsgDecoded

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded
dwpMessageCount	(out) A pointer to a variable in which the number of retrieved messages (actually only 0 or 1) is returned
dwpLostMessageCount	(out) A pointer to a variable in which the host buffer's current number of lost messages is returned
wMessageLocationAndRemoval	(in) The location in the stack or host buffer of the message to read, and removal instructions (sitalMessageLocationAndRemoval_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_Message_DecodeRaw, sitalBcMaximum_MESSAGE_SIZE, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMessageLocation_NEXT, sitalMessageRemoval_PURGE, sitalMode_BC, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalBc_HostBuffer_Message_GetRaw ( S16BIT     swDevice,
                                    U16BIT *   wapBuffer,
                                    U16BIT     wBufferSize,
                                    U32BIT *   dwpMessageCount,
                                    U32BIT *   dwpLostMessageCount
                                  )
```

Read from the host buffer of given bc device as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the host buffer the messages that were read.

Note:

- Given buffer is first zeroed, and then filled with available messages, where each message is stored as follows:
  - A number of sitalBcMaximum_MESSAGE_SIZE memory words is dedicated per message, never mind its actual size.
  - The data stack pointer is replaced with a word whose:
    - MSByte contains the count of data words for the message that was read.
    - LSByte contains the type of the message that was read (sitalMessageType_*).
  - The data words are stored right after the longest possible message block, that is, starting at offset sitalBcMaximum_ACTUAL_MESSAGE_BLOCK. In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
  - Message data is copied under assumption that the expected number of data words has been actually received, what the caller may verify by checking the block status.

Equivalent DDC definition: aceBCGetHBufMsgsRaw

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wapBuffer	(out) A pointer to a buffer in which the raw messages are stored
wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (>0)
dwpMessageCount	(out) A pointer to a variable in which the number of retrieved messages (>=0) is returned

dwpLostMessageCount (out) A pointer to a variable in which the host buffer's current number of lost messages is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBcMaximum_MESSAGE_SIZE, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalBc_Initialize (S16BIT *swDevice*,
U32BIT *dwOptions*
)

Initialize given device as a BC in accordance with given initialization options. Release any past allocations of device memory.

Note:

- This function assumes that function sitalDevice_Initialize has already been called, initialized given device, and inquired its capabilities. This function then lets the user reinitialize given device with a non default configuration.

Equivalent DDC definition: aceBCConfigure

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

dwOptions (in) Initialization options (An or-ed combination of sitalBcSetupOption_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalBcCounter_COMMANDS, sitalBcCounter_DATA_BLOCKS, sitalBcCounter_FRAMES, sitalBcCounter_GPQ_ENTRIES, sitalBcCounter_MESSAGES, sitalBcSetupOption_ASYNCHRONOUS_HIGH_PRIORITY_MODE, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Zero, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalInterruptRegister2_BC_IRQ2, sitalInterruptRegister2_BC_IRQ3, sitalMode_BC, sitalProcess_SetInterruptServiceRoutine(), sitalRegisterAddress_BC_GENERAL_PURPOSE_QUEUE_POINTER, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalBc_Message_Create ( S16BIT  swDevice,
                         S16BIT  swMessageId,
                         S16BIT  swDataBlockIdForMessage1,
                         U16BIT  wBcControlWordForMessage1,
                         U16BIT  wCommandWord1ForMessage1,
                         U16BIT  wCommandWord2ForMessage1,
                         U16BIT  wGapTimeForMessage1,
                         S16BIT  swDataBlockIdForMessage2,
                         U16BIT  wBcControlWordForMessage2,
                         U16BIT  wCommandWord1ForMessage2,
                         U16BIT  wCommandWord2ForMessage2,
                         U16BIT  wGapTimeForMessage2,
                         U32BIT  dwOptions
                       )
```

Create for given BC device a single/dual message for future use by some command, and write it at a dedicated place in device memory.

Note:

- A single message requires 16 data words of device memory.
- Its address/offset (in word) in device memory must be a number whose 5 LSBits are zeros.
- A dual message is actually a set of two messages that are alternately used by a XQF opcode. In order to support such a use, their addresses/offsets (in word) in device memory must only differ in bit #4. Therefore:
    - The first message is located at an address/offset (in word) in device memory whose 5 LSBits are zeros
    - The second message is located right at the end of the first message.
- A dual message is statically allocated for each message, never mind whether a normal, RT-to-RT, or dual message is actually required. Such a message has enough device memory to define a dual RT-to-RT message, which is the longest message possible. Doing so enables the modification of the message using functions sitalBc_Message_Modify* in any desired way, without having to reallocate device memory for it.
- A data block shouldn't be used with more than a single message, though this function does not prevent it.

Equivalent DDC definition: aceBCMsgCreate

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockIdForMessage1	(in) A unique ID designating the data block of message 1 (0-(sitalBcCounter_DATA_BLOCKS-1))
wBcControlWordForMessage1	(in) The BC control word of message 1 (An or-ed combination of sitalBcControlWord_*)
wCommandWord1ForMessage1	(in) Command word 1 of message 1
wCommandWord2ForMessage1	(in) Command word 2 of message 1
wGapTimeForMessage1	(in) The gap of message 1 (>=0)
swDataBlockIdForMessage2	(in) A unique ID designating the data block of message 2 (0-(sitalBcCounter_DATA_BLOCKS-1))
wBcControlWordForMessage2	(in) The BC control word of message 2 (An or-ed combination of sitalBcControlWord_*)
wCommandWord1ForMessage2	(in) Command word 1 of message 2
wCommandWord2ForMessage2	(in) Command word 2 of message 2
wGapTimeForMessage2	(in) The gap of message 2 (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBcControlWord_MODE, sitalBcCounter_DATA_BLOCKS, sitalBcMessageOption_DUAL_MESSAGE, sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_BC_INVALID_DATA_BLOCK_SIZE, sitalReturnCode_BC_OBJECT_ALREADY_EXISTS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MESSAGE_OPTIONS, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_UNDEFINED_NODE.

S16BIT _DECL
sitalBc_Message_CreateBcToBroadcast (S16BIT *swDevice*,
S16BIT *swMessageId*,
S16BIT *swDataBlockId*,
U16BIT *wReceiverRtSubaddress*,
U16BIT *wWordCount*,
U16BIT *wGapTime*,
U32BIT *dwOptions*
)

Create for given BC device a BC-to-Broadcast message.

Note:

- See further information in the documentation for function sitalBc_Message_Create.

Equivalent DDC definition: aceBCMsgCreateBcst

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wReceiverRtSubaddress	(in) Receiver RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wWordCount	(in) Word count (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sital1553_BROADCAST, sitalBc_Message_Create, sitalBcControlWord_BROADCAST, sitalBcControlWord_RETRY_ENABLED, sitalBcMessageOption_DUAL_MESSAGE, sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalBcStld1553_Command_GetCode,

sitalCommand_RX, sitalReturnCode_INVALID_MESSAGE_OPTIONS, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalBc_Message_CreateBcToOrFromBroadcastMode (S16BIT *swDevice*,
S16BIT *swMessageId*,
S16BIT *swDataBlockId*,
U16BIT *wMessageDirection*,
U16BIT *wModeCommand*,
U16BIT *wGapTime*,
U32BIT *dwOptions*
)

Create for given BC device a BC-to-Broadcast-Mode or a Broadcast-to-BC-Mode message, depending on given RT device-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX, respectively).

Note:

- See further information in the documentation for function sitalBc_Message_Create.

Equivalent DDC definition: aceBCMsgCreateBcstMode

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX)
wModeCommand	(in) Mode command (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sital1553_BROADCAST, sital1553_MODE_CODE1, sitalBc_Message_Create, sitalBcControlWord_BROADCAST, sitalBcControlWord_MODE, sitalBcControlWord_RETRY_ENABLED, sitalBcMessageOption_DUAL_MESSAGE,

sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalBcStld1553_Command_GetCode, sitalReturnCode_INVALID_MESSAGE_OPTIONS, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalBc_Message_CreateBcToOrFromRtMode ( S16BIT   swDevice,
                                         S16BIT   swMessageId,
                                         S16BIT   swDataBlockId,
                                         U16BIT   wRt,
                                         U16BIT   wMessageDirection,
                                         U16BIT   wModeCommand,
                                         U16BIT   wGapTime,
                                         U32BIT   dwOptions
                                       )
```

Create for given BC device a BC-to-RT-Mode or a RT-to-BC-Mode message, depending on given RT device-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX, respectively).

Note:

- See further information in the documentation for function sitalBc_Message_Create.

Equivalent DDC definition: aceBCMsgCreateMode

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wRt	(in) Receiver/transmitter RT (sitalRtAddress_0-sitalRtAddress_31)
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX)
wModeCommand	(in) Mode command (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sital1553_MODE_CODE1, sitalBc_Message_Create, sitalBcControlWord_MODE, sitalBcControlWord_RETRY_ENABLED, sitalBcMessageOption_DUAL_MESSAGE, sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalBcStld1553_Command_GetCode, sitalReturnCode_INVALID_MESSAGE_OPTIONS, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL                    ( S16BIT  swDevice,
sitalBc_Message_CreateBcToRt
                                  S16BIT  swMessageId,
                                  S16BIT  swDataBlockId,
                                  U16BIT  wReceiverRt,
                                  U16BIT  wReceiverRtSubaddress,
                                  U16BIT  wWordCount,
                                  U16BIT  wGapTime,
                                  U32BIT  dwOptions
                                )
```

Create for given BC device a BC-to-RT message.

Note:

- See further information in the documentation for function sitalBc_Message_Create.

Equivalent DDC definition: aceBCMsgCreateBCtoRT

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wReceiverRt	(in) Receiver RT (sitalRtAddress_0-sitalRtAddress_31)
wReceiverRtSubaddress	(in) Receiver RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wWordCount	(in) Word count (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_Message_Create, sitalBcControlWord_RETRY_ENABLED, sitalBcMessageOption_DUAL_MESSAGE, sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalBcStld1553_Command_GetCode, sitalCommand_RX, sitalReturnCode_INVALID_MESSAGE_OPTIONS, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalBc_Message_CreateRtToBc ( S16BIT  swDevice,
                               S16BIT  swMessageId,
                               S16BIT  swDataBlockId,
                               U16BIT  wTransmitterRt,
                               U16BIT  wTransmitterRtSubaddress,
                               U16BIT  wWordCount,
                               U16BIT  wGapTime,
                               U32BIT  dwOptions
                             )
```

Create for given BC device a RT-to-BC message.

Note:

- See further information in the documentation for function sitalBc_Message_Create.

Equivalent DDC definition: aceBCMsgCreateRTtoBC

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wTransmitterRt	(in) Transmitter RT (sitalRtAddress_0-sitalRtAddress_31)
wTransmitterRtSubaddress	(in) Transmitter RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wWordCount	(in) Word count (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_Message_Create, sitalBcControlWord_RETRY_ENABLED, sitalBcMessageOption_DUAL_MESSAGE, sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalBcStld1553_Command_GetCode, sitalCommand_TX, sitalReturnCode_INVALID_MESSAGE_OPTIONS, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalBc_Message_CreateRtToBroadcast (S16BIT *swDevice*,
S16BIT *swMessageId*,
S16BIT *swDataBlockId*,
U16BIT *wReceiverRtSubaddress*,
U16BIT *wWordCount*,
U16BIT *wTransmitterRt*,
U16BIT *wTransmitterRtSubaddress*,
U16BIT *wGapTime*,
U32BIT *dwOptions*
)

Create for given BC device a RT-to-Broadcast message.

Note:

- See further information in the documentation for function sitalBc_Message_Create.

Equivalent DDC definition: aceBCMsgCreateBcstRTtoRT

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wReceiverRtSubaddress	(in) Receiver RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wWordCount	(in) Word count (1-32)
wTransmitterRt	(in) Transmitter RT (sitalRtAddress_0-sitalRtAddress_31)
wTransmitterRtSubaddress	(in) Transmitter RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed

Negative sitalReturnCode_* Error condition or function failed

References sital1553_BROADCAST, sitalBc_Message_Create, sitalBcControlWord_BROADCAST, sitalBcControlWord_RETRY_ENABLED, sitalBcControlWord_RT_TO_RT, sitalBcMessageOption_DUAL_MESSAGE, sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalBcStld1553_Command_GetCode, sitalCommand_RX, sitalCommand_TX, sitalReturnCode_INVALID_MESSAGE_OPTIONS, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalBc_Message_CreateRtToRt ( S16BIT   swDevice,
                               S16BIT   swMessageId,
                               S16BIT   swDataBlockId,
                               U16BIT   wReceiverRt,
                               U16BIT   wReceiverRtSubaddress,
                               U16BIT   wWordCount,
                               U16BIT   wTransmitterRt,
                               U16BIT   wTransmitterRtSubaddress,
                               U16BIT   wGapTime,
                               U32BIT   dwOptions
                             )
```

Create for given BC device a RT-to-RT message.

Note:

- See further information in the documentation for function sitalBc_Message_Create.
- A data block ID is required, though the BC does not send or receive data with this kind of message, as the BC still requires a data block to store within the monitored data.

Equivalent DDC definition: aceBCMsgCreateRTtoRT

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wReceiverRt	(in) Receiver RT (sitalRtAddress_0-sitalRtAddress_31)
wReceiverRtSubaddress	(in) Receiver RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wWordCount	(in) Word count (1-32)
wTransmitterRt	(in) Transmitter RT (sitalRtAddress_0-sitalRtAddress_31)
wTransmitterRtSubaddress	(in) Transmitter RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)

dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_Message_Create, sitalBcControlWord_RETRY_ENABLED, sitalBcControlWord_RT_TO_RT, sitalBcMessageOption_DUAL_MESSAGE, sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalBcStld1553_Command_GetCode, sitalCommand_RX, sitalCommand_TX, sitalReturnCode_INVALID_MESSAGE_OPTIONS, and sitalReturnCode_SUCCESS.

S16BIT _DECL **sitalBc_Message_DecodeRaw**	(S16BIT	*swDevice*,	
	U16BIT *	*wapBuffer*,	
	sitalDecodedMessageStructure *	*dmspDecodedMessage*	
	)		

Decode given message of given BC device into given structure.

Note:

- This function does not really require the logical number of the relevant device, which isn't removed only in order to stay compatible with DDC. (DDC use this parameter to verify the given device is a BC at state READY or RUN, which is a non really required restriction.)
- This function assumes that given buffer is at least sitalBcMaximum_MESSAGE_SIZE words long.
- This function assumes that given buffer has been previously filled by function sitalBc_Message_GetByIdRaw. See the documentation for this function for information on the contents of its returned buffer.
- In case given buffer contains a mode code message with data, this data word is returned as the first (actually the single) data word of the decoded message.

Equivalent DDC definition: aceBCDecodeRawMsg

Parameters:

swDevice	(in) Logical number of device (0-31, unused)
wapBuffer	(in) A pointer to a buffer in which a raw message is stored
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBcBlockStatusWord_GOOD_DATA_BLOCK_RECEIVED, sitalBcBlockStatusWord_NO_RESPONSE, sitalBcMaximum_ACTUAL_MESSAGE_BLOCK, sitalMessageType_BC_TO_RT, sitalMessageType_BROADCAST, sitalMessageType_BROADCAST_MODE_DATA, sitalMessageType_BROADCAST_RT_TO_RT, sitalMessageType_MODE_DATA_RX, sitalMessageType_MODE_DATA_TX, sitalMessageType_MODE_NO_DATA, sitalMessageType_RT_TO_BC, sitalMessageType_RT_TO_RT, sitalReturnCode_INVALID_PARAMETER, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalBc_Message_Delete (S16BIT *swDevice*,
S16BIT *swMessageId*
)

Delete given message which has been previously created for given BC device.

Note:

- A regular message may be principally deleted along frame run in order to be compatible with DDC's library; Yet, delete isn't supported if this message is the message transmitted by any existing message-transmitter command, so that a situation in which a regular command orders the transmission of an already deleted message is prohibited.
- An asynchronous message may be principally deleted along frame run in order to be compatible with DDC's library; Moreover, this must be allowed in order to let the user change the group of asynchronous messages along time while running a frame; Yet, if the user does so right after requesting the Tx of asynchronous message[s], then overrides the device memory that has been originally allocated for the deleted message,- all of this before the device had a chance to transmit this message, it may cause unexpected results.

Equivalent DDC definition: aceBCMsgDelete

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBcCounter_COMMANDS, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalOpcode_EXECUTE_AND_FLIP, sitalOpcode_EXECUTE_MESSAGE, sitalOpcode_TIME_CONDITIONED_MESSAGE_TX, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_NOT_SUPPORTED, sitalReturnCode_SUCCESS, and sitalReturnCode_UNDEFINED_MESSAGE_BLOCK.

S16BIT _DECL **sitalBc_Message_GetByIdDecoded**	(S16BIT		*swDevice,*
	S16BIT		*swMessageId,*
	sitalDecodedMessageStructure *		*dmspDecodedMessage,*
	U16BIT		*wIsPurgeRequired*
	)		

Read given message of given BC device, decode it into given structure, and then purge given message if so requested.

Note:

- In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.

Equivalent DDC definition: aceBCGetMsgFromIDDecoded

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded
wIsPurgeRequired	(in) A flag that says whether should zero the end of message bit of the block status word within the message block of given message

Returns:

1 Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_Message_DecodeRaw, sitalBcMaximum_MESSAGE_SIZE, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalBc_Message_GetByIdRaw (S16BIT *swDevice*,
S16BIT *swMessageId*,
U16BIT * *wapBuffer*,
U16BIT *wIsPurgeRequired*
)

Read given message of given BC device in its raw state into given buffer, and then purge given message if so requested.

Note:

- This function assumes that given buffer is at least sitalBcMaximum_MESSAGE_SIZE words long.
- Given buffer is first zeroed, and then filled as follows:
  - The target message block is copied into it.
  - The data stack pointer is replaced with a word whose:
    - MSByte contains the count of data words for the message that was read.
    - LSByte contains the type of the message that was read (sitalMessageType_*).
  - The data words are stored right after the longest possible message block, that is, starting at offset sitalBcMaximum_ACTUAL_MESSAGE_BLOCK. In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- Message data is copied under assumption that the expected number of data words has been actually received, what the caller may verify by checking the block status.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.

Equivalent DDC definition: aceBCGetMsgFromIDRaw

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
wapBuffer	(out) A pointer to a buffer into which given message is copied
wIsPurgeRequired	(in) A flag that says whether should zero the end of message bit of the block status word within the message block of given message

Returns:

1 Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, and sitalReturnCode_INVALID_STATE.

```
S16BIT _DECL
sitalBc_Message_Modify ( S16BIT  swDevice,
                         S16BIT  swMessageId,
                         S16BIT  swDataBlockIdForMessage1,
                         U16BIT  wBcControlWordForMessage1,
                         U16BIT  wCommandWord1ForMessage1,
                         U16BIT  wCommandWord2ForMessage1,
                         U16BIT  wGapTimeForMessage1,
                         S16BIT  swDataBlockIdForMessage2,
                         U16BIT  wBcControlWordForMessage2,
                         U16BIT  wCommandWord1ForMessage2,
                         U16BIT  wCommandWord2ForMessage2,
                         U16BIT  wGapTimeForMessage2,
                         U16BIT  wModificationFlags
                       )
```

Modify a message that has been previously created for given BC device.

See further information in the documentation for function sitalBc_Message_Create.

Equivalent DDC definition: aceBCMsgModify

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockIdForMessage1	(in) A unique ID designating the data block of message 1 (0-(sitalBcCounter_DATA_BLOCKS-1))
wBcControlWordForMessage1	(in) The BC control word of message 1 (An or-ed combination of sitalBcControlWord_*)
wCommandWord1ForMessage1	(in) Command word 1 of message 1
wCommandWord2ForMessage1	(in) Command word 2 of message 1
wGapTimeForMessage1	(in) The gap of message 1 (>=0)
swDataBlockIdForMessage2	(in) A unique ID designating the data block of message 2 (0-(sitalBcCounter_DATA_BLOCKS-1))
wBcControlWordForMessage2	(in) The BC control word of message 2 (An or-ed combination of sitalBcControlWord_*)

wCommandWord1ForMessage2	(in) Command word 1 of message 2
wCommandWord2ForMessage2	(in) Command word 2 of message 2
wGapTimeForMessage2	(in) The gap of message 2 (>=0)
wModificationFlags	(in) Message modification options (An or-ed combination of sitalBcMessageModifier_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBcControlWord_MODE, sitalBcCounter_DATA_BLOCKS, sitalBcMessageModifier_MESSAGE1_BC_CONTROL, sitalBcMessageModifier_MESSAGE1_COMMAND1, sitalBcMessageModifier_MESSAGE1_COMMAND2, sitalBcMessageModifier_MESSAGE1_DATA_BLOCK, sitalBcMessageModifier_MESSAGE1_GAPTIME, sitalBcMessageModifier_MESSAGE2_BC_CONTROL, sitalBcMessageModifier_MESSAGE2_COMMAND1, sitalBcMessageModifier_MESSAGE2_COMMAND2, sitalBcMessageModifier_MESSAGE2_DATA_BLOCK, sitalBcMessageModifier_MESSAGE2_GAPTIME, sitalBcMessageOption_DUAL_MESSAGE, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalReturnCode_BC_INVALID_DATA_BLOCK_SIZE, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_UNDEFINED_DATA_BLOCK, and sitalReturnCode_UNDEFINED_MESSAGE_BLOCK.

```
S16BIT _DECL
sitalBc_Message_ModifyBcToBroadcast ( S16BIT   swDevice,
                                      S16BIT   swMessageId,
                                      S16BIT   swDataBlockId,
                                      U16BIT   wReceiverRtSubaddress,
                                      U16BIT   wWordCount,
                                      U16BIT   wGapTime,
                                      U32BIT   dwOptions,
                                      U16BIT   wModificationFlags
                                    )
```

Modify a BC-to-Broadcast message that has been previously created for given BC device.

Equivalent DDC definition: aceBCMsgModifyBcst

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wReceiverRtSubaddress	(in) Receiver RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wWordCount	(in) Word count (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)
wModificationFlags	(in) Message modification options (An or-ed combination of sitalBcMessageModifier_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sital1553_BROADCAST, sitalBc_Message_Modify, sitalBcControlWord_BROADCAST, sitalBcControlWord_RETRY_ENABLED, sitalBcMessageOption_DUAL_MESSAGE, sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalBcStld1553_Command_GetCode,

sitalCommand_RX, sitalReturnCode_INVALID_MESSAGE_OPTIONS, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalBc_Message_ModifyBcToOrFromBroadcastMode (S16BIT *swDevice*,
S16BIT *swMessageId*,
S16BIT *swDataBlockId*,
U16BIT *wMessageDirection*,
U16BIT *wModeCommand*,
U16BIT *wGapTime*,
U32BIT *dwOptions*,
U16BIT *wModificationFlags*
)

Modify a BC-to-Broadcast-Mode or Broadcast-to-BC-Mode message that has been previously created for given BC device, depending on given RT device-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX, respectively).

Equivalent DDC definition: aceBCMsgModifyBcstMode

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX)
wModeCommand	(in) Mode command (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)
wModificationFlags	(in) Message modification options (An or-ed combination of sitalBcMessageModifier_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sital1553_BROADCAST, sital1553_MODE_CODE1, sitalBc_Message_Modify, sitalBcControlWord_BROADCAST, sitalBcControlWord_MODE, sitalBcControlWord_RETRY_ENABLED, sitalBcMessageOption_DUAL_MESSAGE,

sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalBcStld1553_Command_GetCode, sitalReturnCode_INVALID_MESSAGE_OPTIONS, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalBc_Message_ModifyBcToOrFromRtMode ( S16BIT  swDevice,
                                         S16BIT  swMessageId,
                                         S16BIT  swDataBlockId,
                                         U16BIT  wRt,
                                         U16BIT  wMessageDirection,
                                         U16BIT  wModeCommand,
                                         U16BIT  wGapTime,
                                         U32BIT  dwOptions,
                                         U16BIT  wModificationFlags
                                       )
```

Modify a BC-to-RT-Mode or RT-to-BC message that has been previously created for given BC device, depending on given RT device-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX, respectively).

Equivalent DDC definition: aceBCMsgModifyMode

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wRt	(in) Receiver/transmitter RT (sitalRtAddress_0-sitalRtAddress_31)
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX)
wModeCommand	(in) Mode command (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)
wModificationFlags	(in) Message modification options (An or-ed combination of sitalBcMessageModifier_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sital1553_MODE_CODE1, sitalBc_Message_Modify, sitalBcControlWord_MODE, sitalBcControlWord_RETRY_ENABLED, sitalBcMessageOption_DUAL_MESSAGE, sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalBcStld1553_Command_GetCode, sitalReturnCode_INVALID_MESSAGE_OPTIONS, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalBc_Message_ModifyBcToRt ( S16BIT   swDevice,
                               S16BIT   swMessageId,
                               S16BIT   swDataBlockId,
                               U16BIT   wReceiverRt,
                               U16BIT   wReceiverRtSubaddress,
                               U16BIT   wWordCount,
                               U16BIT   wGapTime,
                               U32BIT   dwOptions,
                               U16BIT   wModificationFlags
                             )
```

Modify a BC-to-RT message that has been previously created for given BC device.

See further information in the documentation for function sitalBc_Message_Modify.

Equivalent DDC definition: aceBCMsgModifyBCtoRT

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wReceiverRt	(in) Receiver RT (sitalRtAddress_0-sitalRtAddress_31)
wReceiverRtSubaddress	(in) Receiver RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wWordCount	(in) Word count (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)
wModificationFlags	(in) Message modification options (An or-ed combination of sitalBcMessageModifier_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_Message_Modify, sitalBcControlWord_RETRY_ENABLED, sitalBcMessageOption_DUAL_MESSAGE, sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalBcStld1553_Command_GetCode, sitalCommand_RX, sitalReturnCode_INVALID_MESSAGE_OPTIONS, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalBc_Message_ModifyRtToBc (S16BIT *swDevice*,
S16BIT *swMessageId*,
S16BIT *swDataBlockId*,
U16BIT *wTransmitterRt*,
U16BIT *wTransmitterRtSubaddress*,
U16BIT *wWordCount*,
U16BIT *wGapTime*,
U32BIT *dwOptions*,
U16BIT *wModificationFlags*
)

Modify a RT-to-BC message that has been previously created for given BC device.

See further information in the documentation for function sitalBc_Message_Modify.

Equivalent DDC definition: aceBCMsgModifyRTtoBC

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wTransmitterRt	(in) Transmitter RT (sitalRtAddress_0-sitalRtAddress_31)
wTransmitterRtSubaddress	(in) Transmitter RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wWordCount	(in) Word count (1-32)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)
wModificationFlags	(in) Message modification options (An or-ed combination of sitalBcMessageModifier_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_Message_Modify, sitalBcControlWord_RETRY_ENABLED, sitalBcMessageOption_DUAL_MESSAGE, sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalBcStld1553_Command_GetCode, sitalCommand_TX, sitalReturnCode_INVALID_MESSAGE_OPTIONS, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalBc_Message_ModifyRtToBroadcast ( S16BIT   swDevice,
                                      S16BIT   swMessageId,
                                      S16BIT   swDataBlockId,
                                      U16BIT   wReceiverRtSubaddress,
                                      U16BIT   wWordCount,
                                      U16BIT   wTransmitterRt,
                                      U16BIT   wTransmitterRtSubaddress,
                                      U16BIT   wGapTime,
                                      U32BIT   dwOptions,
                                      U16BIT   wModificationFlags
                                    )
```

Modify a RT-to-Broadcast message that has been previously created for given BC device.

Equivalent DDC definition: aceBCMsgModifyBcstRTtoRT

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wReceiverRtSubaddress	(in) Receiver RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wWordCount	(in) Word count (1-32)
wTransmitterRt	(in) Transmitter RT (sitalRtAddress_0-sitalRtAddress_31)
wTransmitterRtSubaddress	(in) Transmitter RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)
wModificationFlags	(in) Message modification options (An or-ed combination of sitalBcMessageModifier_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed

Negative sitalReturnCode_* Error condition or function failed

References sital1553_BROADCAST, sitalBc_Message_Modify, sitalBcControlWord_BROADCAST, sitalBcControlWord_RETRY_ENABLED, sitalBcControlWord_RT_TO_RT, sitalBcMessageOption_DUAL_MESSAGE, sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalBcStld1553_Command_GetCode, sitalCommand_RX, sitalCommand_TX, sitalReturnCode_INVALID_MESSAGE_OPTIONS, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalBc_Message_ModifyRtToRt ( S16BIT  swDevice,
                               S16BIT  swMessageId,
                               S16BIT  swDataBlockId,
                               U16BIT  wReceiverRt,
                               U16BIT  wReceiverRtSubaddress,
                               U16BIT  wWordCount,
                               U16BIT  wTransmitterRt,
                               U16BIT  wTransmitterRtSubaddress,
                               U16BIT  wGapTime,
                               U32BIT  dwOptions,
                               U16BIT  wModificationFlags
                             )
```

Modify a RT-to-RT message that has been previously created for given BC device.

See further information in the documentation for function sitalBc_Message_Modify.

Equivalent DDC definition: aceBCMsgModifyRTtoRT

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swMessageId	(in) A unique ID designating a message (0-(sitalBcCounter_MESSAGES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalBcCounter_DATA_BLOCKS-1))
wReceiverRt	(in) Receiver RT (sitalRtAddress_0-sitalRtAddress_31)
wReceiverRtSubaddress	(in) Receiver RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wWordCount	(in) Word count (1-32)
wTransmitterRt	(in) Transmitter RT (sitalRtAddress_0-sitalRtAddress_31)
wTransmitterRtSubaddress	(in) Transmitter RT's subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wGapTime	(in) Gap time (in units of 1us) between the beginning of the message to the beginning of the following message (>=0)
dwOptions	(in) Message options (An or-ed combination of sitalBcControlWord_* and sitalBcMessageOption_*)

wModificationFlags	(in) Message modification options (An or-ed combination of sitalBcMessageModifier_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_Message_Modify, sitalBcControlWord_RETRY_ENABLED, sitalBcControlWord_RT_TO_RT, sitalBcMessageOption_DUAL_MESSAGE, sitalBcMessageOption_STAY_ON_ALTERNATE_BUS, sitalBcStld1553_Command_GetCode, sitalCommand_RX, sitalCommand_TX, sitalReturnCode_INVALID_MESSAGE_OPTIONS, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalBc_MessageRetryPolicy_Set ( S16BIT  swDevice,
                                 U16BIT  wNumberOfRetries,
                                 U16BIT  wFirstRetryBus,
                                 U16BIT  wSecondRetryBus,
                                 U16BIT  wReserved
                               )
```

Configure the message retry policy of given BC device to given number of retries and given first and second chance bus to retry the message with.

Note:

- In order to make a BC device retry a message in case of error, all the following terms must be matched:
  - Function sitalBc_MessageRetryPolicy_Set has been called to configure the BC device with a suitable number of retries and actually available retry buses.
  - Each of the messages that should be retried on error has been defined with option sitalBcControlWord_RETRY_ENABLED upon its creation using function sitalBc_Message_Create* (or similarly modified after creation using function italBc_Message_Modify*).

Equivalent DDC definition: aceBCSetMsgRetry

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wNumberOfRetries	(in) Desired number of retries on message failure (sitalMessageRetryCount_*)
wFirstRetryBus	(in) The bus to retry the message with in the first time in case of message failure (sitalMessageRetryBus_*)
wSecondRetryBus	(in) The bus to retry the message with in the second time in case of message failure (sitalMessageRetryBus_*)
wReserved	(in) Reserved for future use

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister1_BC_DOUBLE_RETRY_ENABLED, sitalConfigurationRegister1_BC_RETRY_MASK, sitalConfigurationRegister1_BC_SINGLE_RETRY_ENABLED,

sitalConfigurationRegister4_BC_RETRY_BUS_1, sitalConfigurationRegister4_BC_RETRY_BUS_2, sitalDeviceState_READY, sitalMessageRetryBus_SAME, sitalMessageRetryCount_NONE, sitalMessageRetryCount_ONE, sitalMode_BC, sitalRegisterAddress_CONFIGURATION_1, sitalRegisterAddress_CONFIGURATION_4, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL **sitalBc_Start** (S16BIT *swDevice*,
S16BIT *swFrameId*,
S32BIT *sdwFrameCount*
)

Make given BC device start running given major frame for given number of times.

Note:

- A major frame may only be run directly using function sitalBc_Start.
- A major frame may not be contained within any frame, neither within itself, nor within another major frame, nor within any minor frame.
- A major frame may contain nested minor frames, and each contained minor frame may in its turn contain other minor frames.
- A minor frame may not be contained within itself, either directly or indirectly. Here are examples of forbidden frame nesting stacks:
    - Major frame 1 >> Minor frame 2 >> Minor frame 2
    - Major frame 1 >> Minor frame 2 >> Minor frame 3 >> Minor frame 2 In order to verify that this rule isn't overridden by the caller, a tracking nesting stack is used, which is a stack that tracks the state of frame nesting. In other words, this stack is properly updated all along the process of building the requested major frame in device memory in order to always reflect the current stack of nested frames. For example, if major frame 1 contains minor frame 2, and minor frame 2 contains minor frame 3, then when reaching minor frame 3 the nesting stack will be (1, 2, 3). A demand imposed by the device is that any moment along this process the size of the nesting stack may not exceed sitalBcMaximum_FRAME_NESTING_STACK.
- Though frame recursion is prevented using the above described mechanism, the caller still has to be careful not to create endless loops in its code.
- When a major frame is being created in device memory, special framing commands are added to those defined for it by the user;
  Here's the framing scheme for major frames:
  * The numbers at the left are logical word addresses.
  * The 'F'-s at the left designate framing commands.
  * The command is executed always where no condition is specified.
  F 00 - Focus-variable No.8
  F 01 - Load-focused-variable with the LSWord of iterations count
  F 02 - Focus-variable No.9
  F 03 - Load-focused-variable with the MSWord of iterations count
  04 - Perform the series of commands assigned for this frame by the user
  * 04 is the body of the loop that's performed by the major frame.

F 05 - Jump always(if running forever)-or-never(if running 1 or more times) to-address-04
F 06 - Focus-variable No.8
F 07 - Decrement-focused-variable by-1
F 08 - Compare-focused-variable to-zero
F 09 - Jump if-not-equal to-address-04
F 10 - Focus-variable No.9
F 11 - Compare-focused-variable to-zero
F 12 - Jump if-equal to-address-15
F 13 - Decrement-focused-variable by-1
F 14 - Jump to-address-04
F 15 - Push-to-general-purpose-queue major-frame-ended-flag
F 16 - Push-to-general-purpose-queue major-frame-ID
F 17 - Halt

These framing commands may be divided into groups as follows:

- Group A - Loop commands (14):
  - Commands that make the major frame repeat itself for the requested number of times (one or more, or even forever).
  - Relevant: Always.
- Group B - Report commands (2):
  - Commands that are aimed at reporting the user via the GPQ about the end of another iteration of the major frame.
  - Relevant: Always.
- Group C - A closing halt command (1):
  - Relevant: Always. These sum up to a total of 17 commands. NOTICE: As may be seen above, hardware variables 8 & 9 are used by this library for framing major frames; They shouldn't, therefore, be used for any other purpose.

- When a minor frame is being created in device memory, special framing commands are added to those defined for it by the user.
  Here's the framing scheme for minor frames:
  * The numbers at the left are logical word addresses.
  * The 'F'-s at the left designate framing commands.
  * The command is executed always where no condition is specified.
  * Some of the framing commands are required only in certain conditions. Nevertheless, for the sake of simplicity and uniformity all framing commands are always added, just that suitable conditions are added.
  F 00 - Load-frame-time
  F 01 - Start-frame-time
  02 - Perform the series of commands assigned for this frame by the user
  * Above mentioned series is usually a series of message execution commands.
  F 03 - Call if-GPF-7-is-on(if low priority asynchronous message Tx is supported)-or-

never(otherwise) the-low-priority-asynchronous-message-Tx-subroutine
F 04 - Push-to-general-purpose-queue minor-frame-ended-flag
F 05 - Push-to-general-purpose-queue minor-frame-ID
F 06 - Issue-interrupt always(if interrupts are supported with given minor frame)-or-never(otherwise) IRQ-3
F 07 - Compare-frame-time always(if high priority asynchronous message Tx is supported)-or-never(otherwise) to-expected-time
F 08 - Push-to-general-purpose-queue if-greater-than(if high priority asynchronous message Tx is supported)-or-never(otherwise) frame-time-overflow-flag
F 09 - Push-to-general-purpose-queue if-greater-than(if high priority asynchronous message Tx is supported)-or-never(otherwise) minor-frame-ID
F 10 - Compare-frame-time to-expected-time
* 10 starts a loop whose aim is to spend the required amount of passive time at the end of the current minor frame.
F 11 - Call if-GPF-6-is-on(if high priority asynchronous message Tx is supported)-or-never(otherwise) the-high-priority-asynchronous-message-Tx-subroutine
F 12 - Jump if-less-than to-address-10
F 13 - Issue-interrupt if-good-message(if both high priority asynchronous message Tx is supported and host buffer is assigned)-or-never(otherwise) IRQ-2
F 14 - Return

These framing commands may be divided into groups as follows:

- Group A - Frame time measuring commands (4):
  - Commands that load the requested frame time, measure the active phase of the frame, and complete it with long enough passive phase.
  - Relevant: Always.
- Group B - Report commands (2):
  - Commands that are aimed at reporting the user via the GPQ about the end of another iteration of the minor frame and of the possibly following low priority asynchronous messages.
  - Relevant: Always.
- Group C - A closing return command (1):
  - Relevant: Always.
- Group D - IRQ-3 interrupt command (1): To sign the end of another iteration of the minor frame (including its active phase alone) together with the low (not high!) priority asynchronous messages that possibly follow this minor frame.
  - Relevant: Always (Indeed, interrupt is only issued in case IRQs aren't disabled for either this minor frame or its containing major frame, but as the containing major frame and its configuration become known only once frame run is requested, and in order to determine already upon the creation of a frame its overall size in device memory, this group will be always included, just that in cases where interrupts aren't enabled, the relevant commands will be conditioned 'never').

    - Group E - Low priority asynchronous messages Tx subroutine calling command (1):
        - Relevant: If low priority asynchronous messages are supported.
    - Group F - Time overflow report commands (3):
        - Commands that are aimed at reporting the user via the GPQ about frame time overflow.
        - Relevant: If high priority asynchronous messages are supported.
    - Group G - High priority asynchronous message Tx subroutine calling command (1):
        - To transmit the high priority asynchronous message along the passive phase.
        - Relevant: If high priority asynchronous messages are supported.
    - Group H - IRQ-2 interrupt command (1):
        - To sign the end of another iteration of the minor frame (including both its active and passive phase) together with the low and high priority asynchronous messages that possibly follow this minor frame.
        - Relevant: If high priority asynchronous messages are supported (recall that the end of the minor frame itself and of the possibly following low priority asynchronous messages has already been signed using IRQ-3), and a host buffer has been assigned to given device (this final interrupt is merely used to update this host buffer). These sum up to a total of 7 to 14 truly relevant commands.
- Calling the subroutines that perform the high & low priority asynchronous message Tx (including their initial stubs) is conditioned by GPF-6 & GPF-7, respectively, being on; User applications should restrict themselves to never use GPF-6 and GPF-7!
- The subroutines that transmit asynchronous messages are both common to all the frames that take part in running given major frame. With the subroutine that's dedicated to low priority asynchronous messages, the count of included asynchronous messages is anyway only determined at the moment when their transmission is actually requested. For these reasons, these subroutines aren't part of the major frame, nor of any of the nested minor frames.
- Once function sitalBc_Start is called for some BC device, function sitalBc_Stop must be called in order to stop that BC, even if it already completed the transmission of the requested number of frame.

Equivalent DDC definition: aceBCStart

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swFrameId	(in) A unique ID designating a frame (0-(sitalBcCounter_FRAMES-1))

sdwFrameCount	(in) The number of times given frame should be transmitted (zero also means single Tx), or sitalBcFrameIterations_RUN_FOREVER if the frame should be run forever (>=-1)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalBc_Command_GetWord(), sitalBcCounter_DATA_BLOCKS, sitalBcCounter_FRAMES, sitalBcCounter_MESSAGES, sitalBcFrameType_MAJOR, sitalBcSetupOption_ASYNCHRONOUS_HIGH_PRIORITY_MODE, sitalBcSetupOption_ASYNCHRONOUS_LOW_PRIORITY_MODE, sitalConfigurationRegister3_ENHANCED_MODE, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalModeVariant_NO_TIME_TAG_RESET, sitalOpcode_EXECUTE_AND_FLIP, sitalOpcode_RETURN_FROM_SUBROUTINE, sitalOpcodeCondition_ALWAYS, sitalOpcodeCondition_GOOD_MESSAGE, sitalRegisterAddress_BC_GENERAL_PURPOSE_QUEUE_POINTER, sitalRegisterAddress_BC_INITIAL_INSTRUCTION_POINTER, sitalRegisterAddress_CONFIGURATION_3, sitalRegisterAddress_START_OR_RESET, sitalReturnCode_ALLOCATION_FAIL, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_NOT_MAJOR_FRAME, sitalReturnCode_SUCCESS, sitalReturnCode_UNDEFINED_NODE, sitalStartResetRegister_BC_MT_START, and sitalStartResetRegister_RESET.

S16BIT _DECL **sitalBc_Stop** (S16BIT *swDevice*)

Stop given BC device from transmitting frames, either at the end of the current message or at the end of the current frame (what comes first). In case of a failure to do so, reset given device.

Note:

- Once function sitalBc_Start is called for some BC device, function sitalBc_Stop must be called in order to stop that BC, even if it already completed the transmission of the requested number of frame.

Equivalent DDC definition: aceBCStop

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister1_BC_BUSY, sitalConfigurationRegister6_BC_ENHANCED, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_BC, sitalRegisterAddress_CONFIGURATION_1, sitalRegisterAddress_CONFIGURATION_6, sitalRegisterAddress_START_OR_RESET, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, sitalStartResetRegister_BC_MT_STOP_AT_END_OF_MESSAGE, and sitalStartResetRegister_BC_STOP_AT_END_OF_FRAME.

S16BIT _DECL
sitalBcStld1553_Command_GetCode (U16BIT * *wpCommandWord*,
U16BIT *wRtAddress*,
U16BIT *wMessageDirection*,
U16BIT *wSubaddressOrMode*,
U16BIT *wWordCountOrModeCode*
)

Create a suitable IEEE-1553 command word based on given parameters.

Equivalent DDC definition: aceCmdWordCreate

Parameters:

wpCommandWord	(out) A pointer to a variable in which requested command word is returned
wRtAddress	(in) A RT address (sitalRtAddress_0-sitalRtAddress_31 or sital1553_BROADCAST)
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX)
wSubaddressOrMode	(in) Subaddress/mode selector (subaddress: sitalRtSubaddress_1-sitalRtSubaddress_30; mode: sital1553_MODE_CODE1/sital1553_MODE_CODE2)
wWordCountOrModeCode	(in) Word-count/mode-word, depending whether regular/mode command, respectively (word-count: 1-32; mode-word: 1-32)
wWordCount	(in) Word count (1-32)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalReturnCode_INVALID_DIRECTION_BIT, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_RT_ADDRESS, sitalReturnCode_INVALID_SUBADDRESS_OR_MODE_SELECTOR, sitalReturnCode_INVALID_WORD_COUNT_OR_MODE_CODE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalBcStld1553_Command_Parse (U16BIT *wCommandWord*,
U16BIT * *wpRtAddress*,
U16BIT * *wpMessageDirection*,
U16BIT * *wpSubaddressOrMode*,
U16BIT * *wpWordCountOrModeCode*
)

Parse given IEEE-1553 command word.

Equivalent DDC definition: aceCmdWordParse

Parameters:

wCommandWord	(in) Command word
wpRtAddress	(out) A pointer to a variable in which the parsed RT address is returned as sitalRtAddress_0-sitalRtAddress_31
wpMessageDirection	(out) A pointer to a variable in which the parsed RT-related message direction is returned as sitalMessageDirection_RX or sitalMessageDirection_TX
wpSubaddressOrMode	(out) A pointer to a variable in which the parsed subaddress/mode selector is returned as sitalRtSubaddress_0-sitalRtSubaddress_31
wpWordCountOrModeCode	(out) A pointer to a variable in which the parsed word-count/mode-word, depending whether regular/mode command, respectively, is returned as 0-31

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalReturnCode_INVALID_PARAMETER, and sitalReturnCode_SUCCESS.

S16BIT _DECL sitalDevice_Clock_SetFrequency (S16BIT *swDevice*,
U16BIT *wClockFrequency*
)

Set the clock frequency of given device to be the given frequency.

Equivalent DDC definition: aceSetClockFreq

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalClockSetup_10MHZ, sitalClockSetup_12MHZ, sitalClockSetup_16MHZ, sitalClockSetup_20MHZ, sitalConfigurationRegister5_CLOCK_SELECT_12MHZ, sitalConfigurationRegister5_CLOCK_SELECT_16MHZ, sitalConfigurationRegister5_CLOCK_SELECT_MASK, sitalConfigurationRegister6_CLOCK_SELECT_10MHZ, sitalConfigurationRegister6_CLOCK_SELECT_12MHZ, sitalConfigurationRegister6_CLOCK_SELECT_16MHZ, sitalConfigurationRegister6_CLOCK_SELECT_20MHZ, sitalConfigurationRegister6_CLOCK_SELECT_MASK, sitalDeviceState_READY, sitalModeVariant_ADVANCED, sitalRegisterAddress_CONFIGURATION_5, sitalRegisterAddress_CONFIGURATION_6, sitalReturnCode_INVALID_CLOCK_FREQUENCY, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_STATE, sitalReturnCode_NOT_SUPPORTED, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalDevice_ConfigureDecoder (S16BIT *swDevice*,
U16BIT *wDecodedInput*,
U16BIT *wExpendedXingOption*
)

Configure the Manchester-II decoder.

Equivalent DDC definition: aceSetDecoderConfig

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wDecodedInput	(in) The type of input to decode (sitalInputDevice_*)
wExpendedXingOption	(in) The sampling method (sitalExpendedXing_*)

Returns:

sitalReturnCode_SUCCESS	Function successfully completed
Negative sitalReturnCode_*	Error condition or function failed

References sitalConfigurationRegister5_EXPANDED_CROSSING, sitalConfigurationRegister5_SINGLE_ENDED_SELECT, sitalDeviceState_READY, sitalExpendedXing_Enable, sitalInputDevice_DoubleEnded, sitalModeVariant_ADVANCED, sitalRegisterAddress_CONFIGURATION_5, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_STATE, sitalReturnCode_NOT_SUPPORTED, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalDevice_ConfigureRamParityCheck (S16BIT *swDevice*,
U16BIT *wRamParityCheckEnabler*
)

Enable/disable RAM parity checking for hardware containing 17-bit buffered RAM.

Equivalent DDC definition: aceSetRamParityChecking

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRamParityCheckEnabler	(in) Automatic RAM parity checking enabler (sitalRamParityCheck_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister2_RAM_PARITY, sitalDeviceState_READY, sitalRamParityCheck_ENABLE, sitalRegisterAddress_CONFIGURATION_2, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalDevice_ConfigureWatchdogTimeout (S16BIT *swDevice*,
U16BIT *bIsWatchdogEnabled*,
U16BIT *wWatchdogTimeout*
)

Configure the watchdog timeout for given device, that is, either enable and set it to given timeout, or disable it, as requested.

Note:

- Parameter wWatchdogTimeout is used only is the watchdog timeout is enabled.

Equivalent DDC definition: aceBCSetWatchDogTimer

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
bIsWatchdogEnabled	(in) A flag that says if the watchdog should be enabled or not
wWatchdogTimeout	(in) The watchdog timeout (in units of 100us) to set for given device (>=0)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister7_BC_ENHANCED_WATCHDOG_TIMER_ENABLE, sitalDeviceState_READY, sitalMode_BC, sitalRegisterAddress_CONFIGURATION_7, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalDevice_CoreConfiguration_Get (S16BIT *swDevice*,
U16BIT * *wpCoreConfiguration*
)

Get the core configuration of given device.

Equivalent DDC definition: aceGetTimeTagValue

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wpCoreConfiguration	(out) A pointer to a variable in which the core configuration is returned (An or-ed combination of sitalCoreConfiguration_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalCoreConfiguration_LIMITED_OPERATION, sitalDeviceState_READY, sitalDeviceState_RUN, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalDevice_Free (S16BIT *swDevice*)

Reset and free given device.

Note:

- This library avoids using any dynamic memory allocations, and this function therefore has no memory allocations to release.
- Once this function successfully completes, given device is reset, and must be reinitialized before any further use.

Equivalent DDC definition: aceFree

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_Close(), and sitalReturnCode_SUCCESS.

S16BIT _DECL sitalDevice_Initialize (S16BIT *swDevice*,
U16BIT *wAccess*,
U16BIT *wMode*,
U32BIT *dwSizeOfAllocatedMemory*,
U32BIT *dwRegistersAddress*,
U32BIT *dwMemoryAddress*
)

Initialize hardware & software resources (i.e., memory and register space) of given device for a given mode of operation. Access modes:

- Card memory: The card of target device is accessed using the device driver.
- Simulated memory: A 64K or 4K chunk of host memory is allocated and manipulated as if it were hardware memory. In this mode the user can produce a binary image file, but cannot actually run a frame. This mode isn't supported currently.
- User memory: Memory and register addresses are passed to the library. This mode isn't supported currently.

Note:

- This function may be also used to reinitialize a device. In such case all the data blocks, messages, commands, and frames that have been previously defined for this device will be deleted.
- Given device is initialized in given mode, but with default options. In order to configure the device with specifically required options, after this function successfully completes, call the suitable sital*_Initialize function, where '*' = Bc/Rt/Mt/RtMt, according to given mode.
- In order for a device to be supported, the total size of its memory must stand the following conditions:
    - BC, RT, MT: Between sitalMinimum_SIZE_OF_DEVICE_MEMORY and sitalMaximum_SIZE_OF_DEVICE_MEMORY.
    - RT&MT: Between sitalMinimum_SIZE_OF_RT_AND_MT_DEVICE_MEMORY and sitalMaximum_SIZE_OF_DEVICE_MEMORY.
    - Any kind of device: A product of sitalMinimum_SIZE_OF_DEVICE_MEMORY by a power of 2.
- This function sets given device to the following default configuration:
    - All devices (sitalMode_*):
        - Interrupts: sitalIrqMode_LEVEL, sitalIrqClear_NO_AUTO_CLEAR
        - Clock frequency: sitalClockSetup_16MHZ
        - Decoder: sitalInputDevice_DoubleEnded, sitalExpendedXing_Enable
        - Response timeout: sitalResponseTimeout_18US
        - Time tag resolution: sitalTimeTagResolution_2US
        - RAM parity check: sitalRamParityCheck_DISABLE
    - RT devices (sitalMode_RT):
        - Messages are illegalized for all address type, direction, subaddress, and word-count-or-mode-code combinations. To legalize a specific combination, either of the following functions may be used, as appropriate: sitalRt_DataBlock_MapToSubaddress, sitalRt_MessageLegality_Enable.
        - Size of command stack (in words) vs. total size of device memory (in words): >=65536/2048, >=32768/1024, >=16384/512, >=2048/256
    - MT devices (sitalMode_MT):
        - Message monitoring enabled for: sitalRtAddress_ALL, sitalMessageDirection_BOTH, sitalRtSubaddressMask_ALL.
        - Stacks:
            - Single pair of stacks, i.e., a single data stack and a single command stack (sitalMtStackOption_SINGLE).
            - Size of data stack (in words) vs. total size of device memory (in words): >=65536/32768, >=32768/16384, >=16384/8192, >=8192/4096, >=4096/2048, >=2048/1024

        - Size of command stack (in words) vs. total size of device memory (in words): >=65536/16384, >=16384/4096, >=8192/1024, >=4096/1024, >=2048/256
    - RT&MT devices (sitalMode_RT_AND_MT):
        - Messages are illegalized for all address type, direction, subaddress, and word-count-or-mode-code combinations. To legalize a specific combination, either of the following functions may be used, as appropriate: sitalRt_DataBlock_MapToSubaddress, sitalRt_MessageLegality_Enable.
        - Message monitoring enabled for: sitalRtAddress_ALL, sitalMessageDirection_BOTH, sitalRtSubaddressMask_ALL.
        - Stacks:
            - Size of RT command stack (in words) vs. total size of device memory (in words): >=65536/2048, >=32768/1024, >=16384/512, >=2048/256
            - Single pair of MT stacks, i.e., a single MT data stack and a single MT command stack (sitalMtStackOption_SINGLE).
            - Size of MT data stack (in words) vs. total size of device memory (in words): >=65536/32768, >=32768/16384, >=16384/8192, >=8192/4096, >=4096/1024
            - Size of MT command stack (in words) vs. total size of device memory (in words): >=65536/16384, >=16384/4096, >=8192/1024, >=4096/256

Equivalent DDC definition: aceInitialize

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wAccess	(in) Type of device access (sitalAccess_*)
wMode	(in) Operation mode (sitalMode_*)
dwSizeOfAllocatedMemory	(in) Size of memory (unused)
dwRegistersAddress	(in) Base address for registers (unused)
dwMemoryAddress	(in) Base address for memory (unused)

Returns:

sitalReturnCode_SUCCESS	Function successfully completed
Negative sitalReturnCode_*	Error condition or function failed

References sitalAccess_CARD, sitalBc_HostBuffer_Free, sitalBc_Initialize, sitalDevice_AccessMemory(), sitalDevice_Free, sitalDevice_Open(), sitalDevice_SetMode(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Registers, sitalDeviceState_READY,

sitalLimitationsRegister_LIMITED_PERFORMANCE,
sitalMaximum_SIZE_OF_DEVICE_MEMORY,
sitalMinimum_SIZE_OF_DEVICE_MEMORY,
sitalMinimum_SIZE_OF_RT_AND_MT_DEVICE_MEMORY, sitalMode_BC, sitalMode_MT,
sitalMode_RT, sitalMode_RT_AND_MT, sitalMode_TEST, sitalModeVariant_ADVANCED,
sitalModeVariant_NONE, sitalMt_HostBuffer_Free, sitalMt_Initialize,
sitalMtCommandStackSize_1024, sitalMtCommandStackSize_16384,
sitalMtCommandStackSize_256, sitalMtCommandStackSize_4096, sitalMtDataStackSize_1024,
sitalMtDataStackSize_16384, sitalMtDataStackSize_2048, sitalMtDataStackSize_32768,
sitalMtDataStackSize_4096, sitalMtDataStackSize_512, sitalMtDataStackSize_8192,
sitalMtStackOption_SINGLE, sitalRegisterAddress_BORDER,
sitalRegisterAddress_LIMITATIONS, sitalRegisterAddress_START_OR_RESET,
sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER,
sitalReturnCode_INVALID_MEMORY_SIZE, sitalReturnCode_INVALID_MODE,
sitalReturnCode_INVALID_MODE_OPTIONS, sitalReturnCode_SUCCESS,
sitalRt_HostBuffer_Free, sitalRt_Initialize, sitalRtCommandStackSize_1024,
sitalRtCommandStackSize_2048, sitalRtCommandStackSize_256,
sitalRtCommandStackSize_512, sitalRtMt_HostBuffer_Free, sitalRtMt_Initialize, and
sitalStartResetRegister_RESET.

S16BIT _DECL
sitalDevice_Irq_Configure (S16BIT *swDevice*,
U16BIT *wInterruptMode*,
U16BIT *wAutoClear*
)

Configure interrupt-related behavior (i.e., type of signal and post-read auto-clearing of interrupt status) for given device.

Note:

- This library sets the interrupt mode to the proper value, level or pulse, depending on the operating-system and the card type upon device initialization. The user is nevertheless allowed to use this function in order to change the interrupt mode, but such an operation may make the interrupts function in an improper way.
- In order for the user to change one of the two modes, the interrupt mode and the interrupt auto-clear mode, without changing the other, he/she may query the current modes using function sitalDevice_Irq_GetMode.

Equivalent DDC definition: aceSetIrqConfig

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wInterruptMode	(in) Requested configuration for type of interrupt signal (sitalIrqMode_*)
wAutoClear	(in) Requested configuration for post-read auto-clearing of interrupt status (sitalIrqClear_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister2_ENHANCED_INTERRUPTS, sitalConfigurationRegister2_INTERRUPT_STATUS_AUTO_CLEAR, sitalConfigurationRegister2_LEVEL_OR_PULSE_INTERRUPTS, sitalDeviceState_READY, sitalIrqClear_AUTO_CLEAR, sitalIrqMode_LEVEL, sitalRegisterAddress_CONFIGURATION_2, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL sitalDevice_Irq_GetMode (S16BIT *swDevice*,
U16BIT * *wpInterruptMode*,

U16BIT * *wpAutoClear*
)

Get the currently configured interrupt-related behavior (i.e., type of signal and post-read auto-clearing of interrupt status) for given device.

Note:

- See the documentation for function sitalDevice_Irq_Configure.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wpInterruptMode	(in) A pointer to a variable within which current type of interrupt signal (sitalIrqMode_*) is returned
wpAutoClear	(in) A pointer to a variable within which current configuration for post-read auto-clearing of interrupt status (sitalIrqClear_*) is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_PARAMETER, and sitalReturnCode_SUCCESS.

S16BIT _DECL **sitalDevice_Irq_Manipulate** (S16BIT *swDevice*,
U16BIT *bIsInterruptEnabled*,
U32BIT *dwIrqMask*,
void(_DECL *funcpExternalIsr)(S16BIT swDevice, U32BIT dwIrqStatus)
)

Enable/disable interrupt service routine calls in case of given interrupts for given device.

Note:

- In order to receive notifications whenever a minor frame completes, the caller should use this function to enable sitalInterruptRegister2_BC_IRQ3 interrupts.
- The effect of arguments bIsInterruptEnabled and dwIrqMask is independent of that of argument funcpExternalIsr, and vice versa.

Equivalent DDC definition: aceSetIrqConditions

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
bIsInterruptEnabled	(in) A flag indicating whether each of the interrupts designated by parameter dwIrqMask should be enabled (if TRUE) or, otherwise, disabled (if FALSE)
dwIrqMask	(in) A mask designating a set of interrupts that should be enabled/disabled as indicated by parameter bIsInterruptEnabled (An or-ed combination of sitalInterruptRegister*)
funcpExternalIsr	(in) A pointer to an interrupt service routine to use with given device, or NULL to stop the usage of a formerly assigned interrupt service routine

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalDevice_Isq_Clear (S16BIT *swDevice*)

Clear the ISQ, and reset the ISQ pointer REGISTER.

Equivalent DDC definition: aceISQClear

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Zero, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_MT, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_RT_MT_INTERRUPT_STATUS_QUEUE_POINTER, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_ISQ_DISABLED, sitalReturnCode_NOT_SUPPORTED, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalDevice_Isq_Configure (S16BIT *swDevice*,
U16BIT *bIsIsqEnabled*
)

Enable/disable the interrupt status queue.

Equivalent DDC definition: aceISQEnable

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
bIsIsqEnabled	(in) A flag that says whether the ISQ should be enabled, or otherwise disabled (TRUE/FALSE)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister6_RT_MT_INTERRUPT_STATUS_QUEUE, sitalDeviceState_READY, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_CONFIGURATION_6, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_NOT_SUPPORTED, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalDevice_Isq_ReadEntry (S16BIT *swDevice*,
sitalIsqEntryStructure * *iespIsqEntry*
)

Read the oldest yet unread entry of the interrupt status queue, and track ISQ overruns.

Equivalent DDC definition: aceISQRead

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))
iespIsqEntry (out) A pointer to the structure in which the ISQ entry is returned

Returns:

sitalReturnCode_ReadIsqEntry_NO_NEW_ONES No entries were read
sitalReturnCode_ReadIsqEntry_READ_NEW_ENTRY One entry was read
sitalReturnCode_ReadIsqEntry_READ_NEW_ENTRY_AND_DETECTED_OVERRUN One entry was read, and one other or more were lost
Negative sitalReturnCode_* Error condition or function failed

References sitalBcCounter_ISQ_ENTRIES, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Zero, sitalDeviceMemorySection_Ram, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_MT, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_RT_MT_INTERRUPT_STATUS_QUEUE_POINTER, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_ISQ_DISABLED, sitalReturnCode_NOT_SUPPORTED, sitalReturnCode_ReadIsqEntry_NO_NEW_ONES, sitalReturnCode_ReadIsqEntry_READ_NEW_ENTRY, sitalReturnCode_ReadIsqEntry_READ_NEW_ENTRY_AND_DETECTED_OVERRUN, and sitalReturnCode_SUCCESS.

U16BIT _DECL
sitalDevice_Memory_Read (S16BIT *swDevice*,
U16BIT *wDeviceMemoryAddress*
)

Read and return the current value of the device memory word at given address for given device.

Note:

- In order to stay compatible with DDC, this function returns zero in case it encounters any kind of problem (bad parameters, any irrelevancy, operation failure). That's though this way a caller, in case zero is returned, can't realize whether the target register really contains zero, or may be some problem occurred.

Equivalent DDC definition: aceMemRead

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wDeviceMemoryAddress	(in) The address of the target memory word in the memory section of the device

Returns:

0x0000-0xFFFF The current value of the given device memory word
0 Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalModeVariant_ADVANCED, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalDevice_Memory_Write (S16BIT *swDevice*,
U16BIT *wDeviceMemoryAddress*,
U16BIT *wDeviceMemoryValue*
)

Write given value into the memory word at given address for given device.

Equivalent DDC definition: aceMemWrite

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wDeviceMemoryAddress	(in) The address of the target memory word in the memory section of the device
wDeviceMemoryValue	(in) The value to set for given memory word

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalModeVariant_ADVANCED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL **sitalDevice_OperationalStatisticsCollection_Configure**	(	S16BIT	*swDevice*,
		U16BIT	*bIsOperationalStatisticsCollected*
	)		

Configure whether operational statistics shall be collected for given device. These operational statistics include host buffer, stack, and GPQ fullness statistics.

Note:

- Collected operational statistics isn't reset in case operational statistics collection is turned on/off. Collected operational statistics simply aren't updated while collection is off.

Equivalent DDC definition: aceSetMetrics

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
bIsOperationalStatisticsCollected	(in) A flag that says whether operational statistics should be collected with this device

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalMode_TEST, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

U16BIT _DECL
sitalDevice_Register_Read (S16BIT *swDevice*,
U16BIT *wRegisterAddress*
)

Read and return the current value of the register at given address for given device.

Note:

- This function actually reads given device register, not from its host image.
- In order to stay compatible with DDC, this function returns zero in case it encounters any kind of problem (bad parameters, any irrelevancy, operation failure). That's though this way a caller, in case zero is returned, can't realize whether the target register really contains zero, or may be some problem occurred.
- In case given device that wasn't initialized with mode-variant sitalModeVariant_ADVANCED, given register won't be read unless given device is currently in state sitalDeviceState_READY.

Equivalent DDC definition: aceRegRead

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRegisterAddress	(in) The address of the target register in the registers section of the device

Returns:

0x0000-0xFFFF The current value of the given register
0 Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalModeVariant_ADVANCED, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalDevice_Register_UpdateDevice (S16BIT *swDevice*,
U16BIT *wRegisterAddress*
)

Use the current image of the registers to actually update the registers of given device.

Note:

- Besides for some exceptional cases, this library actually updates device register only when a frame is run. Therefore, in situations where no frame run takes place, the registers aren't really written to the device, even in case they were properly configured using this library functions. This function is aimed at solving this in such situations by forcing the actual update of given device register with its currently configured value.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRegisterAddress	(in) The address of the target register in the registers section of the device

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalModeVariant_ADVANCED, sitalRegisterAddress_CONFIGURATION_2, sitalReturnCode_INVALID_DEVICE_NUMBER, and sitalReturnCode_INVALID_STATE.

S16BIT _DECL
sitalDevice_Register_Write (S16BIT *swDevice*,
U16BIT *wRegisterAddress*,
U16BIT *wRegisterValue*
)

Write given value into the register at given address for given device.

Note:

- If given device is currently in state sitalDeviceState_READY, this function writes given value into given register's host image, not into the real device register. If given device is currently in state sitalDeviceState_RUN, this function actually writes given value into given device register, not into its host image.
- In case given device that wasn't initialized with mode-variant sitalModeVariant_ADVANCED, given register won't be written unless given device is currently in state sitalDeviceState_READY.

Equivalent DDC definition: aceRegWrite

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRegisterAddress	(in) The address of the target register in the registers section of the device
wRegisterValue	(in) The value to set for given register

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalModeVariant_ADVANCED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalDevice_SetAsynchronousMessagesIsr (S16BIT *swDevice*,
void(_DECL *funcpAsynchronousMessagesIsr)(S16BIT swDevice, U16BIT wMinorFrameId)
)

Set the interrupt service routine to use with given device in case some high priority asynchronous message violates the time frame of a minor frame.

Equivalent DDC definition: aceSetAsyncIsr

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
funcpAsynchronousMessagesIsr	(in) A pointer to an asynchronous messages interrupt service routine to use with given device, or NULL to stop the usage of an asynchronous messages interrupt service routine

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalAccess_USER, sitalDeviceState_READY, sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalDevice_SetResponseTimeout (S16BIT *swDevice*,
U16BIT *wResponseTimeout*
)

Set given response timeout for given device.

Equivalent DDC definition: aceSetRespTimeOut

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

wResponseTimeout (in) Response timeout (sitalResponseTimeout_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed

Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister5_RESPONSE_TIMEOUT_130US, sitalConfigurationRegister5_RESPONSE_TIMEOUT_18US, sitalConfigurationRegister5_RESPONSE_TIMEOUT_22US, sitalConfigurationRegister5_RESPONSE_TIMEOUT_50US, sitalConfigurationRegister5_RESPONSE_TIMEOUT_MASK, sitalDeviceState_READY, sitalRegisterAddress_CONFIGURATION_5, sitalResponseTimeout_18US, sitalResponseTimeout_22US, sitalResponseTimeout_50US, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalDevice_Test_Interrupts (S16BIT *swDevice*,
sitalDeviceTestResultStructure * *dtrspTestResult*
)

Reset given device and perform device interrupts test. This test consists of the following steps:

- Set the device time tag resolution to test mode: Set bits 7-9 of its Configuration Register #2 to 011(2).
- Set the device to use level type interrupts: Set bit 3 of its Configuration Register #2 to 1(2).
- Configure the device to issue time tag rollover interrupts: Set bit 6 of its Interrupt Mask Register #1 to 1(2).
- Assign an ISR to the device that will record the IRQ status of informed interrupts.
- Make the device generate a time tag rollover interrupt: Load its time tag register with a value of 0xFFFF, and then increment it to force a time tag rollover.
- Verify that the expected interrupt has indeed been tracked by the assigned ISR.

Note:

- The following check-IDs are returned by this function in case of test failures:
  - 1: The time tag register was not reset though a rollover condition has been artificially created.
  - 2: A time tag rollover interrupt was not issued.

Equivalent DDC definition: aceTestIrqs

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dtrspTestResult (in) A pointer to a structure within which the test result is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalConfigurationRegister2_INTERRUPT_STATUS_AUTO_CLEAR, sitalConfigurationRegister2_LEVEL_INTERRUPTS, sitalConfigurationRegister2_TIME_TAG_TEST, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceTestResult_INTERRUPT_FAIL, sitalDeviceTestResult_PASS, sitalInterruptMaskRegister1_TIME_TAG_ROLLOVER,

sitalInterruptRegister1_MASTER_INTERRUPT, sitalInterruptRegister1_TIME_TAG_ROLLOVER, sitalMode_TEST, sitalProcess_GetInterruptLatency(), sitalProcess_Log_PrintLine(), sitalProcess_SetInterruptServiceRoutine(), sitalRegisterAddress_CONFIGURATION_2, sitalRegisterAddress_INTERRUPT_MASK_1, sitalRegisterAddress_START_OR_RESET, sitalRegisterAddress_TIME_TAG, sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_INVALID_TEST_STRUCTURE, sitalReturnCode_SUCCESS, sitalStartResetRegister_RESET, and sitalStartResetRegister_TIME_TAG_TEST_CLOCK.

S16BIT _DECL
sitalDevice_Test_Memory (S16BIT *swDevice*,
sitalDeviceTestResultStructure * *dtrspTestResult*,
U16BIT *wWrittenValue*
)

Reset given device and perform device memory test. Fill all the memory of given device with given value, and verify that the written values may be read back from the device.

Note:

- The following check-IDs are returned by this function in case of test failures:
  - i: The memory address whose r/w test failed.

Equivalent DDC definition: aceTestMemory

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dtrspTestResult (in) A pointer to a structure within which the test result is returned
wWrittenValue (in) A value to fill device memory with.

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceTestResult_PASS, sitalDeviceTestResult_RAM_FAIL, sitalMode_TEST, sitalRegisterAddress_START_OR_RESET, sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_INVALID_TEST_STRUCTURE, sitalReturnCode_SUCCESS, and sitalStartResetRegister_RESET.

S16BIT _DECL
sitalDevice_Test_Protocol (S16BIT *swDevice*,
sitalDeviceTestResultStructure * *dtrspTestResult*
)

References sitalAccess_CARD, sitalBcBlockStatusWord_BUS_B, sitalBcBlockStatusWord_END_OF_MESSAGE, sitalBcBlockStatusWord_ERROR_FLAG, sitalBcBlockStatusWord_FORMAT_ERROR, sitalBcBlockStatusWord_LOOPBACK_FAIL, sitalBcBlockStatusWord_NO_RESPONSE, sitalBcBlockStatusWord_START_OF_MESSAGE, sitalBcBlockStatusWord_STATUS_SET, sitalConfigurationRegister1_BC_FRAME_AUTOREPEAT, sitalConfigurationRegister1_BC_FRAME_STOP_ON_ERROR, sitalConfigurationRegister1_BC_STATUS_SET_STOP_ON_FRAME, sitalConfigurationRegister1_BC_STATUS_SET_STOP_ON_MESSAGE, sitalConfigurationRegister2_TIME_TAG_2US, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceTestResult_PASS, sitalDeviceTestResult_PROTOCOL_FAIL, sitalMode_TEST, sitalRegisterAddress_CONFIGURATION_1, sitalRegisterAddress_CONFIGURATION_2, sitalRegisterAddress_START_OR_RESET, sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_INVALID_TEST_STRUCTURE, sitalReturnCode_SUCCESS, sitalReturnCode_TEST_FAIL, sitalStartResetRegister_BC_MT_START, and sitalStartResetRegister_RESET.

S16BIT _DECL **sitalDevice_Test_Registers** (S16BIT *swDevice*,
sitalDeviceTestResultStructure * *dtrspTestResult*
)

Reset given device and perform device registers test. This test consists of the following steps:

- Loop over a predetermined set of tested device registers, and with each tested register verify that:
  - It properly responds r/w operations.
  - It is reset upon device reset.
- Verify the correct operation of the time tag system.

Note:

- The following check-IDs are returned by this function in case of test failures:
  - i: Either the index of the faulty register test iteration, or the index of the faulty time tag register test iteration.

Equivalent DDC definition: aceTestRegisters

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dtrspTestResult (in) A pointer to a structure within which the test result is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalConfigurationRegister2_TIME_TAG_TEST, sitalConfigurationRegister3_ENHANCED_MODE, sitalConfigurationRegister4_RT_LATCH_ADDRESS, sitalConfigurationRegister4_TEST_MODE_REGISTER, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceTestResult_PASS, sitalDeviceTestResult_REGISTER_FAIL, sitalMode_TEST, sitalRegisterAddress_CONFIGURATION_2, sitalRegisterAddress_CONFIGURATION_3, sitalRegisterAddress_CONFIGURATION_4, sitalRegisterAddress_INTERRUPT_MASK_1, sitalRegisterAddress_START_OR_RESET, sitalRegisterAddress_TIME_TAG, sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_INVALID_TEST_STRUCTURE, sitalReturnCode_SUCCESS,

sitalStartResetRegister_RESET, sitalStartResetRegister_TIME_TAG_RESET, and sitalStartResetRegister_TIME_TAG_TEST_CLOCK.

S16BIT _DECL **sitalDevice_Test_Vectors** (S16BIT *swDevice*,
sitalDeviceTestResultStructure * *dtrspTestResult*,
char * *szpVectorsFilePath*
)

Reset given device and perform device vector test. All the test vectors that are available in given vector file will be retrieved one at a time, and applied to given device hardware. After applying all these vectors, registers and memory will be read to ensure the test of given vector group passed indeed.

Note:

- In case given device has been initialized as:
  - A test device: It must be in state "ready", and a h/w reset will be performed before starting the test.
  - An operative device (BC, RT, MT, or RT&MT): It must be in either state "ready" or "run", and a h/w reset won't be performed before starting the test.
- A line in the vector file is ignored unless beginning with a 'V' and describing a legal vector.
- A legal vector is made of the following components:
  - A character that designates the type test operation, read ('R') or write ('W'). A test operation of type "write" does no test: It simply writes a desired value into a specific address in prepare to a following "read" test operation. A test operation of type "read" verifies that the expected value is held in a specific address.
  - A character that designates the target section of the test operation, registers ('R') or RAM ('M').
  - A hexadecimal word (a 4-digit zero-padded number) that designates the target address in the target section.
  - A hexadecimal word (a 4-digit zero-padded number) that designates in case of a test of type:
    - "write": The value to write to the target address in the target section.
    - "read": The value expected to be read from the target address in the target section.
- Here are sample legal vector lines:
  - "V W R 0000 FFFF"
  - "V R R 0000 FFFF"
  - "V W M 1234 ABCD"
  - "V R M 1234 ABCD"
- Though any line not beginning with 'V' is ignored, the caller is urged to comment-out lines using a starting 'D'.
- The following check-IDs are returned by this function:

- In case of test failures: The number (1, 2, 3, ...) of the line in given vectors file whose test failed.
- In case all vector tests succeeded: The total number of lines in given file.

Equivalent DDC definition: aceTestVectors

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dtrspTestResult	(in) A pointer to a structure within which the test result is returned
szpVectorsFilePath	(in) A pointer to a null-terminated string that contains the path of the vector file to use for the test

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalDeviceTestResult_MEMORY_READ_VECTOR_FAIL, sitalDeviceTestResult_PASS, sitalDeviceTestResult_REGISTER_READ_VECTOR_FAIL, sitalDeviceVectorTest_MAXIMUM_NUMBER_OF_CHARACTERS_IN_A_LINE_OF_THE_VECTORS_FILE, sitalMode_TEST, sitalRegisterAddress_START_OR_RESET, sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_FILE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_INVALID_TEST_STRUCTURE, sitalReturnCode_SUCCESS, and sitalStartResetRegister_RESET.

S16BIT _DECL
sitalDevice_TimeTag_Get (S16BIT *swDevice*,
U16BIT * *wpTimeTag*
)

Get the current value of the time tag register.

Equivalent DDC definition: aceGetTimeTagValue

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wpTimeTag	(out) A pointer to a variable in which the current value of the time tag register is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Registers, sitalRegisterAddress_TIME_TAG, sitalReturnCode_INVALID_DEVICE_NUMBER, and sitalReturnCode_SUCCESS.

S16BIT _DECL sitalDevice_TimeTag_Reset (S16BIT *swDevice*)

Reset the time tag register.

Equivalent DDC definition: aceResetTimeTag

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Registers, sitalRegisterAddress_START_OR_RESET, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_SUCCESS, and sitalStartResetRegister_TIME_TAG_RESET.

S16BIT _DECL
sitalDevice_TimeTag_Set (S16BIT *swDevice*,
U16BIT *wTimeTag*
)

Set the time tag register to given value.

Equivalent DDC definition: aceSetTimeTagValue

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

wTimeTag (in) The desired value to set the time tag register to

Returns:

sitalReturnCode_SUCCESS Function successfully completed

Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Registers, sitalRegisterAddress_TIME_TAG, sitalReturnCode_INVALID_DEVICE_NUMBER, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalDevice_TimeTag_SetResolution (S16BIT *swDevice*,
U16BIT *wTimeTagResolution*
)

Set the resolution of the time tag register.

Equivalent DDC definition: aceSetTimeTagRes

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wTimeTagResolution	(in) The desired resolution for the time tag register (sitalTimeTagResolution_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister2_TIME_TAG_16US, sitalConfigurationRegister2_TIME_TAG_2US, sitalConfigurationRegister2_TIME_TAG_32US, sitalConfigurationRegister2_TIME_TAG_4US, sitalConfigurationRegister2_TIME_TAG_64US, sitalConfigurationRegister2_TIME_TAG_8US, sitalConfigurationRegister2_TIME_TAG_EXTERNAL_CLOCK, sitalConfigurationRegister2_TIME_TAG_RESOLUTION_MASK, sitalConfigurationRegister2_TIME_TAG_TEST, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalModeVariant_NO_TIME_TAG_RESET, sitalRegisterAddress_CONFIGURATION_2, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, sitalTimeTagResolution_16US, sitalTimeTagResolution_2US, sitalTimeTagResolution_32US, sitalTimeTagResolution_4US, sitalTimeTagResolution_64US, sitalTimeTagResolution_8US, sitalTimeTagResolution_EXTERNAL_CLOCK, and sitalTimeTagResolution_TEST.

S16BIT _DECL
sitalHoo9_Mt_HostBuffer_Message_GetDecoded (S16BIT *swDevice*,
sitalDecodedMessageStructure * *dmspDecodedMessage*,
U32BIT * *dwpMessageCount*,
U32BIT * *dwpStackLostMessageCount*,
U32BIT * *dwpHostBufferLostMessageCount*,
U16BIT *wMessageLocationAndRemoval*
)

Read from the host buffer of given HOO9 MT device the message at given location, decode it into given structure, and purge it if so required. Also get the number of retrieved messages (actually only 0 or 1), the host buffer's current number of lost messages, and the current number of lost messages for both given MT device's MT stacks.

Note:

- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded
dwpMessageCount	(out) A pointer to a variable in which the number of retrieved messages (actually only 0 or 1) is returned
dwpStackLostMessageCount	(out) A pointer to a variable in which given device stack's current number of lost messages is returned
dwpHostBufferLostMessageCount	(out) A pointer to a variable in which the host buffer's current number of lost messages is returned
wMessageLocationAndRemoval	(in) The location in the stack or host buffer of the message to read, and removal instructions (sitalMessageLocationAndRemoval_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalMt_HostBuffer_Message_GetDecoded.

S16BIT _DECL
sitalHoo9_Mt_HostBuffer_Message_GetRaw (S16BIT *swDevice*,
U16BIT * *wapBuffer*,
U16BIT *wBufferSize*,
U32BIT * *dwpMessageCount*,
U32BIT * *dwpStackLostMessageCount*,
U32BIT * *dwpHostBufferLostMessageCount*
)

Read from the host buffer of given HOO9 MT device as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the host buffer the messages that were read.

Note:

- Given buffer is first zeroed, and then filled with available messages, where each message is stored as follows:
  - A number of sitalMtMaximum_MESSAGE_SIZE memory words is dedicated per message, never mind its actual size.
  - The target MT stack entry is copied into given buffer.
  - The data stack pointer is replaced with a word whose:
    - MSByte contains the count of data words for the message that was read.
    - LSByte contains the type of the message that was read (sitalMessageType_*).
  - The data words are stored right after the stack entry, that is, starting at offset sitalMtMemoryObjectSize_COMMAND_STACK_ENTRY. In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.
- The status-related fields of the sitalDecodedMessageStructure are filled with dummy values, as no statuses are returned in the HOO9 protocol.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wapBuffer	(out) A pointer to a buffer in which the raw messages are stored

wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (>0)
dwpMessageCount	(out) A pointer to a variable in which the number of retrieved messages (>=0) is returned
dwpStackLostMessageCount	(out) A pointer to a variable in which given device stack's current number of lost messages is returned
dwpHostBufferLostMessageCount	(out) A pointer to a variable in which the host buffer's current number of lost messages is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalMt_HostBuffer_Message_GetRaw, and sitalMtMaximum_MESSAGE_SIZE.

S16BIT _DECL **sitalHoo9_Mt_Message_DecodeRaw**	(S16BIT	*swDevice*,
	U16BIT *	*wapBuffer*,
	sitalDecodedMessageStructure *	*dmspDecodedMessage*
	)	

Decode given message of given HOO9 MT device into given structure.

Note:

- This function does not really require the logical number of the relevant device, which isn't removed only in order to stay compatible with DDC. (See the documentation for the corresponding 1553 function, sitalMt_Message_DecodeRaw, which is DDC compatible.)
- This function assumes that given buffer is at least sitalMtMaximum_MESSAGE_SIZE words long.
- This function assumes that given buffer has been previously filled by function sitalHoo9_Mt_Message_GetFromStackRaw. See the documentation for this function for information on the contents of its returned buffer.
- The status-related fields of the sitalDecodedMessageStructure are filled with dummy values, as no statuses are returned in the HOO9 protocol.

Parameters:

swDevice	(in) Logical number of device (0-31, unused)
wapBuffer	(in) A pointer to a buffer in which a raw message is stored
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalMt_Message_DecodeRaw.

S16BIT _DECL
sitalHoo9_Mt_Message_GetFromStackDecoded (S16BIT *swDevice*,
sitalDecodedMessageStructure * *dmspDecodedMessage*,
U16BIT *wMessageLocationAndRemoval*,
U16BIT *wStackSelector*
)

Read from given stack of given HOO9 MT device the message at given location, decode it into given structure, and purge it if so required.

Note:

- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded
wMessageLocationAndRemoval	(in) The location in the stack or host buffer of the message to read, and removal instructions (sitalMessageLocationAndRemoval_*)
wStackSelector	(in) A selector that specifies the stack to read from (sitalMtStack_*)

Returns:

One (1) The [single] requested message was read and decoded
Zero (0) No message available, though no error occurred
Negative sitalReturnCode_* Error condition or function failed

References sitalMt_Message_GetFromStackDecoded.

S16BIT _DECL
sitalHoo9_Mt_Message_GetFromStackRaw (S16BIT *swDevice*,
U16BIT * *wapBuffer*,
U16BIT *wBufferSize*,
U16BIT *wStackSelector*
)

Read from given stack of given HOO9 MT device into given buffer as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the stack the messages that were read.

Note:

- Given buffer is first zeroed, and then filled with available messages, where each message is stored as follows:
  - A number of sitalMtMaximum_MESSAGE_SIZE memory words is dedicated per message, never mind its actual size.
  - The target MT stack entry is copied into given buffer.
  - The data stack pointer is replaced with a word whose:
    - LSByte contains the type of the message that was read (sitalMessageType_*).
    - The MSbit (bit #7) of the MSByte is set in case an error has been discovered with the data words.
    - The rest of the bits (bits #0-6) of the MSByte contain the count of data words of given message.
  - The data words are stored right after the stack entry, that is, starting at offset sitalMtMemoryObjectSize_COMMAND_STACK_ENTRY. In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.
- The status-related fields of the sitalDecodedMessageStructure are filled with dummy values, as no statuses are returned in the HOO9 protocol.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wapBuffer	(out) A pointer to a buffer in which the raw messages are stored
wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (>=sitalMtMaximum_MESSAGE_SIZE)

wStackSelector (in) A selector that specifies the stack to read from (sitalMtStack_*)

Returns:

Non-negative integer The number of messages that were read
Negative sitalReturnCode_* Error condition or function failed

References sitalMt_Message_GetFromStackRaw, and sitalMtMaximum_MESSAGE_SIZE.

S16BIT _DECL
sitalHoo9_Mt_MessageMonitoring_Disable (S16BIT *swDevice*,
U16BIT *wRtAddress*,
U16BIT *wMessageDirection*,
U64BIT *qwRtSubaddressMask*
)

Configure given HOO9 MT device to avoid monitoring commands that suits given combinations of RT address, RT-related message direction, and subaddress.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRtAddress	(in) A RT address (sitalRtAddress_0-sitalRtAddress_15 or sitalRtAddress_ALL)
wMessageDirection	(in) RT-related message direction/bus (sitalMessageDirection_*)
qwRtSubaddressMask	(in) Mask of affected subaddresses, 0-63, where bit #i corresponds subaddress #i (An or-ed combination of sitalHoo9RtSubaddressMask_0-sitalHoo9RtSubaddressMask_63)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

S16BIT _DECL
sitalHoo9_Mt_MessageMonitoring_Enable (S16BIT *swDevice*,
U16BIT *wRtAddress*,
U16BIT *wMessageDirection*,
U64BIT *qwRtSubaddressMask*
)

Configure given HOO9 MT device to monitor commands that suits given combinations of RT address, RT-related message direction, and subaddress.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRtAddress	(in) A RT address (sitalRtAddress_0-sitalRtAddress_15 or sitalRtAddress_ALL)
wMessageDirection	(in) RT-related message direction/bus (sitalMessageDirection_*)
qwRtSubaddressMask	(in) Mask of affected subaddresses, 0-63, where bit #i corresponds subaddress #i (An or-ed combination of sitalHoo9RtSubaddressMask_0-sitalHoo9RtSubaddressMask_63)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

S16BIT _DECL
sitalHoo9_Mt_MessageMonitoring_GetStatus (S16BIT *swDevice*,
U16BIT *wRtAddress*,
U16BIT *wMessageDirection*,
U64BIT * *qwpRtSubaddressMask*
)

Get for given HOO9 MT device and combination of RT address and RT-related message direction a mask specifying the monitored subaddresses.

Note:

- In case no subaddress is monitored for given RT device address and direction, the returned subaddress mask will be zero.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRtAddress	(in) A RT address (sitalRtAddress_0-sitalRtAddress_31)
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX)
qwpRtSubaddressMask	(out) A pointer to a variable within which a mask of monitored subaddresses, 0-63, where bit #i corresponds subaddress #i, is returned, what forms an or-ed combination of sitalHoo9RtSubaddressMask_*

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalMessageDirection_RX, sitalMessageDirection_TX, sitalMt_MessageMonitoring_GetStatus, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_RT_ADDRESS, sitalReturnCode_SUCCESS, and sitalRtAddress_16.

S16BIT _DECL
sitalMt_Continue (S16BIT *swDevice*)

Make given MT device continue capturing messages after a temporary pause. The message capturing activity will continue from the same internal state as at the moment of pausing.

Equivalent DDC definition: aceMTContinue

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalAccess_USER, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Registers, sitalDeviceState_RUN, sitalMode_MT, sitalRegisterAddress_START_OR_RESET, sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalStartResetRegister_BC_MT_START.

S16BIT _DECL
sitalMt_GetInformation (S16BIT *swDevice*,
sitalMtInformationStructure * *mispMtInformation*
)

Return information about the configuration of given MT device.

Equivalent DDC definition: aceMTGetInfo

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
mispMtInformation	(out) A pointer to a structure within which the required configuration information is written

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_MT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalMt_HostBuffer_Free (S16BIT *swDevice*)

Free given MT device's host buffer.

Note:

- See the documentation of function sitalRtMt_HostBuffer_Initialize for more information on using functions sitalRt_HostBuffer_* or sitalMt_HostBuffer_* combined with functions sitalRtMt_HostBuffer_*.

Equivalent DDC definition: aceMTUninstallHBuf

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalDeviceState_READY, sitalInterruptRegister1_BC_RT_COMMAND_STACK_ROLLOVER, sitalInterruptRegister1_END_OF_MESSAGE, sitalInterruptRegister1_MT_COMMAND_STACK_ROLLOVER, sitalInterruptRegister1_MT_DATA_STACK_ROLLOVER, sitalInterruptRegister1_TIME_TAG_ROLLOVER, sitalInterruptRegister2_MT_COMMAND_STACK_HALF_ROLLOVER, sitalInterruptRegister2_MT_DATA_STACK_HALF_ROLLOVER, sitalInterruptRegister2_RT_COMMAND_STACK_HALF_ROLLOVER, sitalMode_MT, sitalMode_RT_AND_MT, sitalProcess_SetInterruptServiceRoutine(), sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_HOST_BUFFER_SIZE, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalMt_HostBuffer_GetOperationalStatistics (S16BIT *swDevice*,
sitalHostBufferOperationalStatisticsStructure * *hbosspHostBufferOperationalStatistics*,
U16BIT *bIsResetOfHighestRecordedPercentageRequired*
)

Return performance information about the host buffer of given MT device.

Equivalent DDC definition: aceMTGetHBufMetric

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
hbosspHostBufferOperationalStatistics	(out) A pointer to the host buffer operational statistics structure into which the required operational statistics are written
bIsResetOfHighestRecordedPercentageRequired	(in) A flag that says whether the record of the highest percentage reached by now should be reset

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_MT, sitalMode_RT_AND_MT, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_OPERATIONAL_STATISTICS_NOT_ENABLED, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalMt_HostBuffer_Initialize (S16BIT *swDevice*,
U32BIT *dwHostBufferSize*
)

Initialize (or re-initialize) given MT device's host buffer.

Note:

- Given size of host buffer must be large enough to contain sitalMtMinimum_COMMAND_STACKS_IN_HOST_BUFFER times the number of raw messages that the command stack of given MT device may contain. The number of messages that may be contained by the command stack (N) is its configured size in words divided by sitalMtMemoryObjectSize_COMMAND_STACK_ENTRY. As the maximum possible size of message is allocated for each message in the host buffer, the corresponding size (in words) is (N / sitalMtMaximum_MESSAGE_SIZE). Therefore, given size must be not less than (sitalMtMinimum_COMMAND_STACKS_IN_HOST_BUFFER * (N / sitalMtMaximum_MESSAGE_SIZE)).

Equivalent DDC definition: aceMTInstallHBuf

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dwHostBufferSize	(in) The desired size (in words) of the host buffer (<above-defined-minimum-size-of-RT-host-buffer>-sitalMaximum_SIZE_OF_HOST_BUFFER)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalDeviceState_READY, sitalInterruptRegister1_MT_COMMAND_STACK_ROLLOVER, sitalInterruptRegister1_MT_DATA_STACK_ROLLOVER, sitalInterruptRegister1_TIME_TAG_ROLLOVER, sitalInterruptRegister2_MT_COMMAND_STACK_HALF_ROLLOVER, sitalInterruptRegister2_MT_DATA_STACK_HALF_ROLLOVER, sitalMaximum_SIZE_OF_HOST_BUFFER, sitalMode_MT, sitalMode_RT_AND_MT, sitalMt_HostBuffer_Free, sitalMtMaximum_MESSAGE_SIZE, sitalMtMemoryObjectSize_COMMAND_STACK_ENTRY, sitalMtMinimum_COMMAND_STACKS_IN_HOST_BUFFER, sitalProcess_SetInterruptServiceRoutine(), sitalReturnCode_INVALID_DEVICE_NUMBER,

sitalReturnCode_INVALID_HOST_BUFFER_SIZE, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalMt_HostBuffer_Message_GetCount (S16BIT *swDevice*)

Get the number of messages that are currently available in the host buffer that is assigned with given MT device.

Equivalent DDC definition: aceMTGetHBufMsgCount

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

Non-negative integer The number of messages that are currently available in the host buffer that is assigned with given device

Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_MT, sitalMode_RT_AND_MT, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, and sitalReturnCode_INVALID_STATE.

S16BIT _DECL **sitalMt_HostBuffer_Message_GetDecoded**	(S16BIT		*swDevice*,
		sitalDecodedMessageStructure *	*dmspDecodedMessage*,
		U32BIT *	*dwpMessageCount*,
		U32BIT *	*dwpStackLostMessageCount*,
		U32BIT *	*dwpHostBufferLostMessageCount*,
		U16BIT	*wMessageLocationAndRemoval*
	)		

Read from the host buffer of given MT device the message at given location, decode it into given structure, and purge it if so required. Also get the number of retrieved messages (actually only 0 or 1), the host buffer's current number of lost messages, and the current number of lost messages for both given MT device's MT stacks.

Note:

- In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.

Equivalent DDC definition: aceMTGetStkMsgDecoded

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded
dwpMessageCount	(out) A pointer to a variable in which the number of retrieved messages (actually only 0 or 1) is returned
dwpStackLostMessageCount	(out) A pointer to a variable in which given device stack's current number of lost messages is returned
dwpHostBufferLostMessageCount	(out) A pointer to a variable in which the host buffer's current number of lost messages is returned

wMessageLocationAndRemoval	(in) The location in the stack or host buffer of the message to read, and removal instructions (sitalMessageLocationAndRemoval_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMessageLocation_NEXT, sitalMessageRemoval_PURGE, sitalMode_MT, sitalMode_RT_AND_MT, sitalMt_Message_DecodeRaw, sitalMtMaximum_MESSAGE_SIZE, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalMt_HostBuffer_Message_GetRaw ( S16BIT     swDevice,
                                    U16BIT *   wapBuffer,
                                    U16BIT     wBufferSize,
                                    U32BIT *   dwpMessageCount,
                                    U32BIT *   dwpStackLostMessageCount,
                                    U32BIT *   dwpHostBufferLostMessageCount
                                  )
```

Read from the host buffer of given MT device as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the host buffer the messages that were read.

Note:

- Given buffer is first zeroed, and then filled with available messages, where each message is stored as follows:
  - A number of sitalMtMaximum_MESSAGE_SIZE memory words is dedicated per message, never mind its actual size.
  - The target MT stack entry is copied into given buffer.
  - The data stack pointer is replaced with a word whose:
    - MSByte contains the count of data words for the message that was read.
    - LSByte contains the type of the message that was read (sitalMessageType_*).
  - The data words are stored right after the stack entry, that is, starting at offset sitalMtMemoryObjectSize_COMMAND_STACK_ENTRY. In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.

Equivalent DDC definition: aceMTGetHBufMsgsRaw

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wapBuffer	(out) A pointer to a buffer in which the raw messages are stored

wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (>0)
dwpMessageCount	(out) A pointer to a variable in which the number of retrieved messages (>=0) is returned
dwpStackLostMessageCount	(out) A pointer to a variable in which given device stack's current number of lost messages is returned
dwpHostBufferLostMessageCount	(out) A pointer to a variable in which the host buffer's current number of lost messages is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_MT, sitalMode_RT_AND_MT, sitalMtMaximum_MESSAGE_SIZE, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalMt_HostBuffer_Message_Record (S16BIT *swDevice*)

Record the newly received messages from the active command and data stacks of given MT device into its host buffer.

Note:

- User applications should not principally use this function, as it is automatically called by this library whenever a relevant interrupt occurs. This function has been originally exported only in order to enable working with operating systems that does not support interrupts.
- Messages are recorded in the host buffer only as long that it isn't full, that is, there's no override of old messages by new ones. The user must repeatedly get messages from the host buffer in order to empty it.

Equivalent DDC definition: aceMTStkToHBuf

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_MT, sitalMode_RT_AND_MT, sitalMt_Stack_Swap, sitalMtMaximum_MESSAGE_SIZE, sitalMtMemoryObjectSize_COMMAND_STACK_ENTRY, sitalMtStack_STACK_A, sitalMtStackOption_DOUBLE, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_MT_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_SUCCESS, and sitalRtMaximum_MESSAGE_SIZE.

S16BIT _DECL
sitalMt_Initialize (S16BIT *swDevice*,
U16BIT *wStackMode*,
U16BIT *wCommandStackSize*,
U16BIT *wDataStackSize*,
U32BIT *dwOptions*
)

Initialize given device as a MT in accordance with given initialization options. Release any past allocations of device memory.

Note:

- This function assumes that function sitalDevice_Initialize has already been called, initialized given device, and inquired its capabilities. This function then lets the user reinitialize given device with a non default configuration.
- Both MT stacks, command and data, must be aligned in the device memory to their size (e.g., a 4096 word stack may be placed only at an address of the form 4096xN where N=0,1,2,...).
- In case the caller specifies stack sizes that the memory of given device is too small to contain, the function call will be rejected.

Equivalent DDC definition: aceMTConfigure

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wStackMode	(in) Stack mode (sitalMtStackOption_*)
wCommandStackSize	(in) Size of command stack (sitalMtCommandStackSize_*)
wDataStackSize	(in) Size of data stack (sitalMtDataStackSize_*)
dwOptions	(in) Initialization options (An or-ed combination of sitalMtSetupOption_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Zero, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalMode_MT, sitalProcess_SetInterruptServiceRoutine(), sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE,

sitalReturnCode_MT_INVALID_COMMAND_STACK_SIZE,
sitalReturnCode_MT_INVALID_DATA_STACK_SIZE,
sitalReturnCode_MT_INVALID_STACK_OPTION, and sitalReturnCode_SUCCESS.

S16BIT _DECL **sitalMt_Message_DecodeRaw**	(S16BIT		*swDevice*,
	U16BIT *		*wapBuffer*,
	sitalDecodedMessageStructure *		*dmspDecodedMessage*
	)		

Decode given message of given MT device into given structure.

Note:

- This function does not really require the logical number of the relevant device, which isn't removed only in order to stay compatible with DDC. (DDC use this parameter to verify the given device is a MT at state READY or RUN, which is a non really required restriction.)
- This function assumes that given buffer is at least sitalMtMaximum_MESSAGE_SIZE words long.
- This function assumes that given buffer has been previously filled by function sitalMt_Message_GetFromStackRaw. See the documentation for this function for information on the contents of its returned buffer.
- In case given buffer contains a mode code message with data, this data word is returned as the first (actually the single) data word.
- Unless given message is a RT-to-RT message, then no more than a single RT is involved with it. In such a non-RT-to-RT message that involves a single RT, if this RT isn't currently monitored by the monitoring MT device, the message isn't recorded at all. In case of a RT-to-RT message there are two RTs involved. If both RTs aren't currently monitored by the monitoring MT device, the message isn't recorded at all. If only one of them is currently monitored by the monitoring MT device, the message is recorded in the command and data stacks in a somewhat distorted way. Now, since this function may be called even a long time after given raw message has been recorded, and the list of monitored RTs may have changed in the while, this function has no way to determine if and how given raw message has been distorted. Therefore, this function does not take in account such situations, what entails in an incorrect decoding of raw RT-to-RT messages that have been recorded while any of their involved RTs wasn't monitored by the monitoring MT. This function behaves this way also in order to stay compatible with DDC.

Equivalent DDC definition: aceMTDecodeRawMsg

Parameters:

swDevice	(in) Logical number of device (0-31, unused)
wapBuffer	(in) A pointer to a buffer in which a raw message is stored

dmspDecodedMessage (out) A pointer to a structure into which a message is decoded

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalMessageType_BC_TO_RT, sitalMessageType_BROADCAST, sitalMessageType_BROADCAST_MODE_DATA, sitalMessageType_BROADCAST_MODE_NO_DATA, sitalMessageType_BROADCAST_RT_TO_RT, sitalMessageType_MODE_DATA_RX, sitalMessageType_MODE_DATA_TX, sitalMessageType_MODE_NO_DATA, sitalMessageType_RT_TO_BC, sitalMessageType_RT_TO_RT, sitalMtBlockStatusWord_GOOD_DATA, sitalMtBlockStatusWord_NO_RESPONSE, sitalMtMemoryObjectSize_COMMAND_STACK_ENTRY, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_SUCCESS, and sitalRtBlockStatusWord_RT_TO_RT_COMMAND_ERROR.

S16BIT _DECL
sitalMt_Message_GetFromStackDecoded (S16BIT *swDevice*,
sitalDecodedMessageStructure * *dmspDecodedMessage*,
U16BIT *wMessageLocationAndRemoval*,
U16BIT *wStackSelector*
)

Read from given stack of given MT device the message at given location, decode it into given structure, and purge it if so required.

Note:

- In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.

Equivalent DDC definition: aceMTGetStkMsgDecoded

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded
wMessageLocationAndRemoval	(in) The location in the stack or host buffer of the message to read, and removal instructions (sitalMessageLocationAndRemoval_*)
wStackSelector	(in) A selector that specifies the stack to read from (sitalMtStack_*)

Returns:

One (1) The [single] requested message was read and decoded
Zero (0) No message available, though no error occurred
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister1_MT_ACTIVE_STACK, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Ram,

sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMessageLocation_NEXT, sitalMessageRemoval_PURGE, sitalMode_MT, sitalMode_RT_AND_MT, sitalMt_Message_DecodeRaw, sitalMtMaximum_MESSAGE_SIZE, sitalMtStack_ACTIVE, sitalMtStack_INACTIVE, sitalMtStack_STACK_A, sitalMtStack_STACK_B, sitalMtStackSelector_STACK_A, sitalMtStackSelector_STACK_B, sitalRegisterAddress_CONFIGURATION_1, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_MT_INVALID_STACK_SELECTOR, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalMt_Message_GetFromStackRaw (S16BIT *swDevice*,
U16BIT * *wapBuffer*,
U16BIT *wBufferSize*,
U16BIT *wStackSelector*
)

Read from given stack of given MT device into given buffer as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the stack the messages that were read.

Note:

- Given buffer is first zeroed, and then filled with available messages, where each message is stored as follows:
  - A number of sitalMtMaximum_MESSAGE_SIZE memory words is dedicated per message, never mind its actual size.
  - The target MT stack entry is copied into given buffer.
  - The data stack pointer is replaced with a word whose:
    - LSByte contains the type of the message that was read (sitalMessageType_*).
    - The MSbit (bit #7) of the MSByte is set in case an error has been discovered with the data words.
    - The rest of the bits (bits #0-6) of the MSByte contain the count of data words of given message.
  - The data words are stored right after the stack entry, that is, starting at offset sitalMtMemoryObjectSize_COMMAND_STACK_ENTRY. In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.

Equivalent DDC definition: aceMTGetStkMsgsRaw

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wapBuffer	(out) A pointer to a buffer in which the raw messages are stored
wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (>=sitalMtMaximum_MESSAGE_SIZE)

wStackSelector (in) A selector that specifies the stack to read from (sitalMtStack_*)

Returns:

Non-negative integer The number of messages that were read
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister1_MT_ACTIVE_STACK, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Ram, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_MT, sitalMode_RT_AND_MT, sitalMtMaximum_MESSAGE_SIZE, sitalMtStack_ACTIVE, sitalMtStack_INACTIVE, sitalMtStack_STACK_A, sitalMtStack_STACK_B, sitalMtStackSelector_STACK_A, sitalMtStackSelector_STACK_B, sitalRegisterAddress_CONFIGURATION_1, sitalReturnCode_INVALID_BUFFER, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_MT_INVALID_STACK_SELECTOR, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalMt_MessageMonitoring_Disable (S16BIT *swDevice*,
U16BIT *wRtAddress*,
U16BIT *wMessageDirection*,
U32BIT *dwRtSubaddressMask*
)

Configure given MT device to avoid monitoring commands that suits given combinations of RT address, RT-related message direction/bus, and subaddress.

Equivalent DDC definition: aceMTDisableRTFilter

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRtAddress	(in) A RT address (sitalRtAddress_*)
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_*)
dwRtSubaddressMask	(in) Mask of affected subaddresses, 0-31, where bit #i corresponds subaddress #i (An or-ed combination of sitalRtSubaddressMask_0-sitalRtSubaddressMask_31)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMessageDirection_BOTH, sitalMode_MT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_DIRECTION_BIT, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_RT_ADDRESS, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalRtAddress_ALL.

S16BIT _DECL
sitalMt_MessageMonitoring_Enable (S16BIT *swDevice*,
U16BIT *wRtAddress*,
U16BIT *wMessageDirection*,
U32BIT *dwRtSubaddressMask*
)

Configure given MT device to monitor commands that suits given combinations of RT address, RT-related message direction, and subaddress.

Equivalent DDC definition: aceMTEnableRTFilter

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRtAddress	(in) A RT address (sitalRtAddress_*)
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_*)
dwRtSubaddressMask	(in) Mask of affected subaddresses, 0-31, where bit #i corresponds subaddress #i (An or-ed combination of sitalRtSubaddressMask_0-sitalRtSubaddressMask_31)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMessageDirection_BOTH, sitalMode_MT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_DIRECTION_BIT, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_RT_ADDRESS, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalRtAddress_ALL.

S16BIT _DECL
sitalMt_MessageMonitoring_GetStatus (S16BIT *swDevice*,
U16BIT *wRtAddress*,
U16BIT *wMessageDirection*,
U32BIT * *dwpRtSubaddressMask*
)

Get for given MT device and combination of RT address and RT-related message direction a mask specifying the monitored subaddresses.

Note:

- In case no subaddress is monitored for given RT device address and direction, the returned subaddress mask will be zero.

Equivalent DDC definition: aceMTGetRTFilter

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRtAddress	(in) A RT address (sitalRtAddress_0-sitalRtAddress_31)
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX)
dwpRtSubaddressMask	(out) A pointer to a variable within which a mask of monitored subaddresses, 0-31, where bit #i corresponds subaddress #i, is returned, what forms an or-ed combination of sitalRtSubaddressMask_*

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_MT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_DIRECTION_BIT, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_RT_ADDRESS, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL **sitalMt_Pause** (S16BIT *swDevice*)

Make given MT device temporarily stop capturing messages.

Equivalent DDC definition: aceMTPause

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalAccess_USER, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Registers, sitalDeviceState_RUN, sitalMode_MT, sitalRegisterAddress_START_OR_RESET, sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalStartResetRegister_BC_MT_STOP_AT_END_OF_MESSAGE.

S16BIT _DECL
sitalMt_Stack_GetOperationalStatistics (S16BIT *swDevice*,
sitalStackOperationalStatisticsStructure * *sosspStackOperationalStatistics*,
U16BIT *wStackSelector*,
U16BIT *bIsResetOfHighestRecordedPercentageRequired*
)

Return performance information about the command stack of given MT device.

Equivalent DDC definition: aceMTGetStkMetric

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
sosspStackOperationalStatistics	(out) A pointer to the stack operational statistics structure into which the required operational statistics are written
wStackSelector	(in) A selector that specifies the stack[s] whose information is required (sitalMtStackSelector_*)
bIsResetOfHighestRecordedPercentageRequired	(in) A flag that says whether the record of the highest percentage reached by now should be reset

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_MT, sitalMode_RT_AND_MT, sitalMtStackOption_DOUBLE, sitalMtStackSelector_STACK_A, sitalMtStackSelector_STACK_B, sitalMtStackSelector_STACK_COMBINED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_OPERATIONAL_STATISTICS_NOT_ENABLED, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalMt_Stack_Swap (S16BIT *swDevice*)

Swap between the active and inactive stacks of given MT device.

Equivalent DDC definition: aceMTSwapStks

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalMtActiveStack_* Indication of the new active stack
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister1_MT_ACTIVE_STACK, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_RUN, sitalMode_MT, sitalMtStackOption_DOUBLE, sitalMtStackSelector_STACK_A, sitalMtStackSelector_STACK_B, sitalRegisterAddress_CONFIGURATION_1, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL **sitalMt_Start** (S16BIT *swDevice*)

Make given MT device start capturing messages.

Equivalent DDC definition: aceMTStart

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalAccess_USER, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_MT, sitalRegisterAddress_START_OR_RESET, sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalStartResetRegister_BC_MT_START.

S16BIT _DECL sitalMt_Stop (S16BIT *swDevice*)

Make given MT device stop capturing messages.

Equivalent DDC definition: aceMTStop

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalAccess_USER, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_MT, sitalRegisterAddress_INTERRUPT_MASK_1, sitalRegisterAddress_START_OR_RESET, sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalStartResetRegister_BC_MT_STOP_AT_END_OF_MESSAGE.

S16BIT _DECL
sitalPp194_Mt_HostBuffer_Message_GetDecoded (S16BIT *swDevice*,
sitalDecodedMessageStructure * *dmspDecodedMessage*,
U32BIT * *dwpMessageCount*,
U32BIT * *dwpStackLostMessageCount*,
U32BIT * *dwpHostBufferLostMessageCount*,
U16BIT *wMessageLocationAndRemoval*
)

Read from the host buffer of given PP194 MT device the message at given location, decode it into given structure, and purge it if so required. Also get the number of retrieved messages (actually only 0 or 1), the host buffer's current number of lost messages, and the current number of lost messages for both given MT device's MT stacks.

Note:

- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded
dwpMessageCount	(out) A pointer to a variable in which the number of retrieved messages (actually only 0 or 1) is returned
dwpStackLostMessageCount	(out) A pointer to a variable in which given device stack's current number of lost messages is returned
dwpHostBufferLostMessageCount	(out) A pointer to a variable in which the host buffer's current number of lost messages is returned
wMessageLocationAndRemoval	(in) The location in the stack or host buffer of the message to read, and removal instructions (sitalMessageLocationAndRemoval_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalMt_HostBuffer_Message_GetDecoded.

S16BIT _DECL
sitalPp194_Mt_HostBuffer_Message_GetRaw (S16BIT *swDevice*,
U16BIT * *wapBuffer*,
U16BIT *wBufferSize*,
U32BIT * *dwpMessageCount*,
U32BIT * *dwpStackLostMessageCount*,
U32BIT * *dwpHostBufferLostMessageCount*
)

Read from the host buffer of given PP194 MT device as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the host buffer the messages that were read.

Note:

- Given buffer is first zeroed, and then filled with available messages, where each message is stored as follows:
  - A number of sitalMtMaximum_MESSAGE_SIZE memory words is dedicated per message, never mind its actual size.
  - The target MT stack entry is copied into given buffer.
  - The data stack pointer is replaced with a word whose:
    - MSByte contains the count of data words for the message that was read.
    - LSByte contains the type of the message that was read (sitalMessageType_*).
  - The data words are stored right after the stack entry, that is, starting at offset sitalMtMemoryObjectSize_COMMAND_STACK_ENTRY. In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.
- The status-related fields of the sitalDecodedMessageStructure are filled with dummy values, as no statuses are returned in the PP194 protocol.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wapBuffer	(out) A pointer to a buffer in which the raw messages are stored
wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (>0)
dwpMessageCount	(out) A pointer to a variable in which the number of retrieved messages (>=0) is returned
dwpStackLostMessageCount	(out) A pointer to a variable in which given device stack's current number of lost messages is returned
dwpHostBufferLostMessageCount	(out) A pointer to a variable in which the host buffer's current number of lost messages is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalMt_HostBuffer_Message_GetRaw.

S16BIT _DECL **sitalPp194_Mt_Message_DecodeRaw**	(S16BIT	*swDevice*,
	U16BIT *	*wapBuffer*,
	sitalDecodedMessageStructure *	*dmspDecodedMessage*
	)	

Decode given message of given PP194 MT device into given structure.

Note:

- This function does not really require the logical number of the relevant device, which isn't removed only in order to stay compatible with DDC. (See the documentation for the corresponding 1553 function, sitalMt_Message_DecodeRaw, which is DDC compatible.)
- This function assumes that given buffer is at least sitalMtMaximum_MESSAGE_SIZE words long.
- This function assumes that given buffer has been previously filled by function sitalPp194_Mt_Message_GetFromStackRaw. See the documentation for this function for information on the contents of its returned buffer.
- The status-related fields of the sitalDecodedMessageStructure are filled with dummy values, as no statuses are returned in the PP194 protocol.

Parameters:

swDevice	(in) Logical number of device (0-31, unused)
wapBuffer	(in) A pointer to a buffer in which a raw message is stored
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded

Returns:

sitalReturnCode_SUCCESS	Function successfully completed
Negative sitalReturnCode_*	Error condition or function failed

References sitalMt_Message_DecodeRaw.

S16BIT _DECL

sitalPp194_Mt_Message_GetFromStackDecoded (	S16BIT	*swDevice*,
	sitalDecodedMessageStructure *	*dmspDecodedMessage*,
	U16BIT	*wMessageLocationAndRemoval*,
	U16BIT	*wStackSelector*
)		

Read from given stack of given PP194 MT device the message at given location, decode it into given structure, and purge it if so required.

Note:

- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded
wMessageLocationAndRemoval	(in) The location in the stack or host buffer of the message to read, and removal instructions (sitalMessageLocationAndRemoval_*)
wStackSelector	(in) A selector that specifies the stack to read from (sitalMtStack_*)

Returns:

One (1) The [single] requested message was read and decoded
Zero (0) No message available, though no error occurred
Negative sitalReturnCode_* Error condition or function failed

References sitalMt_Message_GetFromStackDecoded.

```
S16BIT _DECL
sitalPp194_Mt_Message_GetFromStackRaw ( S16BIT     swDevice,
                                        U16BIT *   wapBuffer,
                                        U16BIT     wBufferSize,
                                        U16BIT     wStackSelector
                                      )
```

Read from given stack of given PP194 MT device into given buffer as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the stack the messages that were read.

Note:

- Given buffer is first zeroed, and then filled with available messages, where each message is stored as follows:
  - A number of sitalMtMaximum_MESSAGE_SIZE memory words is dedicated per message, never mind its actual size.
  - The target MT stack entry is copied into given buffer.
  - The data stack pointer is replaced with a word whose:
    - LSByte contains the type of the message that was read (sitalMessageType_*).
    - The MSbit (bit #7) of the MSByte is set in case an error has been discovered with the data words.
    - The rest of the bits (bits #0-6) of the MSByte contain the count of data words of given message.
  - The data words are stored right after the stack entry, that is, starting at offset sitalMtMemoryObjectSize_COMMAND_STACK_ENTRY. In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.
- The status-related fields of the sitalDecodedMessageStructure are filled with dummy values, as no statuses are returned in the PP194 protocol.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wapBuffer	(out) A pointer to a buffer in which the raw messages are stored
wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (>=sitalMtMaximum_MESSAGE_SIZE)

wStackSelector (in) A selector that specifies the stack to read from (sitalMtStack_*)

Returns:

Non-negative integer The number of messages that were read
Negative sitalReturnCode_* Error condition or function failed

References sitalMt_Message_GetFromStackRaw.

S16BIT _DECL
sitalPp194_Mt_MessageMonitoring_Disable (S16BIT *swDevice*,
U16BIT *wRtAddress*,
U16BIT *wMessageDirection*,
U32BIT *dwRtSubaddressMask*
)

Configure given PP194 MT device to avoid monitoring commands that suits given combinations of RT address, RT-related message direction/bus, and subaddress.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRtAddress	(in) A RT address (sitalRtAddress_0-sitalRtAddress_15 or sitalRtAddress_ALL)
wMessageDirection	(in) RT-related message direction/bus (sitalMessageDirection_*)
dwRtSubaddressMask	(in) Mask of affected subaddresses, 0-15, where bit #i corresponds subaddress #i (An or-ed combination of sitalRtSubaddressMask_0-sitalRtSubaddressMask_15)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalMt_MessageMonitoring_Disable, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_RT_ADDRESS, sitalRtAddress_16, sitalRtAddress_ALL, and sitalRtSubaddressMask_ALL.

S16BIT _DECL
sitalPp194_Mt_MessageMonitoring_Enable (S16BIT *swDevice*,
U16BIT *wRtAddress*,
U16BIT *wMessageDirection*,
U32BIT *dwRtSubaddressMask*
)

Configure given PP194 MT device to monitor commands that suits given combinations of RT address, RT-related message direction/bus, and subaddress.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRtAddress	(in) A RT address (sitalRtAddress_0-sitalRtAddress_15 or sitalRtAddress_ALL)
wMessageDirection	(in) RT-related message direction/bus (sitalMessageDirection_*)
dwRtSubaddressMask	(in) Mask of affected subaddresses, 0-15, where bit #i corresponds subaddress #i (An or-ed combination of sitalRtSubaddressMask_0-sitalRtSubaddressMask_15)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalMt_MessageMonitoring_Enable, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_RT_ADDRESS, sitalRtAddress_16, sitalRtAddress_ALL, and sitalRtSubaddressMask_ALL.

S16BIT _DECL
sitalPp194_Mt_MessageMonitoring_GetStatus (S16BIT *swDevice*,
U16BIT *wRtAddress*,
U16BIT *wMessageDirection*,
U32BIT * *dwpRtSubaddressMask*
)

Get for given PP194 MT device and combination of RT address and RT-related message direction/bus a mask specifying the monitored subaddresses.

Note:

- In case no subaddress is monitored for given RT device address and direction, the returned subaddress mask will be zero.

Equivalent DDC definition: aceMTGetRTFilter

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRtAddress	(in) A RT address (sitalRtAddress_0-sitalRtAddress_31)
wMessageDirection	(in) RT-related message direction/bus (sitalMessageDirection_RX or sitalMessageDirection_TX)
dwpRtSubaddressMask	(out) A pointer to a variable within which a mask of monitored subaddresses, 0-15, where bit #i corresponds subaddress #i, is returned, what forms an or-ed combination of sitalRtSubaddressMask_0-sitalRtSubaddressMask_15

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalMt_MessageMonitoring_GetStatus, sitalReturnCode_INVALID_RT_ADDRESS, sitalRtAddress_16, and sitalRtAddress_ALL.

S16BIT _DECL
sitalRt_Address_Get (S16BIT *swDevice*,
U16BIT * *wpRtAddress*
)

Get the RT address of given RT device.

Note:

- Unlike DDC's aceRTGetAddress, this function also allows user applications that didn't initialize a RT device to get its RT address.

Equivalent DDC definition: aceRTGetAddress

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wpRtAddress (out) A pointer to a variable within which the RT address is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister5_RT_ADDRESS_MASK, sitalDevice_AccessMemory(), sitalDevice_GetInformation(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_CONFIGURATION_5, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL sitalRt_Address_Relatch (S16BIT *swDevice*)

Latch the RT address that is currently input to given RT device.

Equivalent DDC definition: aceRTRelatchAddress

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister3_ENHANCED_MODE, sitalConfigurationRegister4_RT_LATCH_ADDRESS, sitalConfigurationRegister5_RT_ADDRESS_MASK, sitalConfigurationRegister5_RT_ADDRESS_PARITY, sitalConfigurationRegister6_RT_ADDRESS_SOURCE, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_CONFIGURATION_3, sitalRegisterAddress_CONFIGURATION_4, sitalRegisterAddress_CONFIGURATION_5, sitalRegisterAddress_CONFIGURATION_6, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalRtAddressSource_EXTERNAL.

S16BIT _DECL
sitalRt_Address_Set (S16BIT *swDevice*,
U16BIT *wRtAddress*
)

Set the RT address of given RT device.

Equivalent DDC definition: aceRTSetAddress

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRtAddress (in) A RT address (sitalRtAddress_0-sitalRtAddress_31)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister3_ENHANCED_MODE, sitalConfigurationRegister4_RT_LATCH_ADDRESS, sitalConfigurationRegister5_RT_ADDRESS_MASK, sitalConfigurationRegister5_RT_ADDRESS_PARITY, sitalConfigurationRegister6_RT_ADDRESS_SOURCE, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_CONFIGURATION_3, sitalRegisterAddress_CONFIGURATION_4, sitalRegisterAddress_CONFIGURATION_5, sitalRegisterAddress_CONFIGURATION_6, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_LIMITED_DEVICE, sitalReturnCode_SUCCESS, sitalReturnCode_WRITE_ERROR, and sitalRtAddress_24.

S16BIT _DECL
sitalRt_AddressSource_Get (S16BIT *swDevice*,
U16BIT * *wpRtAddressSource*
)

Get RT address source of given RT device.

Equivalent DDC definition: aceRTGetAddrSource

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wpRtAddressSource (out) Address source (sitalRtAddressSource_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
other sitalReturnCode_* Function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_CONFIGURATION_6, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRt_AddressSource_Set (S16BIT *swDevice*,
U16BIT *wRtAddressSource*
)

Set RT address source of given RT device to the one given.

Equivalent DDC definition: aceRTSetAddrSource

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRtAddressSource (in) Desired address source (sitalRtAddressSource_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
other sitalReturnCode_* Function failed

References sitalConfigurationRegister6_RT_ADDRESS_SOURCE, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_CONFIGURATION_6, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRt_BuiltInTestWord_Configure (S16BIT *swDevice*,
U16BIT *wWordLocation*,
U16BIT *wPermitOrInihibitIfRtBusy*
)

Configure the built-in test of given RT device to use the word in given source and to inhibit/permit the built-in test if the RT is busy, as required.

Equivalent DDC definition: aceRTBITWrdConfig

Parameters:

swDevice	(in) Logical number of device (0-31, unused)
wWordLocation	(in) The source where to read the built-in test word from (sitalRtBuiltInTestWordSource_*)
wPermitOrInihibitIfRtBusy	(in) A flag that says whether to permit/inhibit the built-in test if the RT is busy (sitalRtBuiltInTestPermitOnBusy_*)

Returns:

sitalReturnCode_SUCCESS	Function successfully completed
Negative sitalReturnCode_*	Error condition or function failed

References sitalConfigurationRegister4_RT_EXTERNAL_BUILT_IN_TEST_WORD, sitalConfigurationRegister4_RT_INHIBIT_BUILT_IN_TEST_WORD_IF_BUSY, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_CONFIGURATION_4, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, sitalRtBuiltInTestPermitOnBusy_ENABLED, and sitalRtBuiltInTestWordSource_REGISTER.

S16BIT _DECL
sitalRt_BuiltInTestWord_Read (S16BIT *swDevice*,
U16BIT *wWordLocation*,
U16BIT * *wpWord*
)

Read the built-in test word of given RT device that is stored at given location.

Equivalent DDC definition: aceRTBITWrdRead

Parameters:

swDevice	(in) Logical number of device (0-31, unused)
wWordLocation	(in) The source where to read the built-in test word from (sitalRtBuiltInTestWordSource_*)
wpWord	(out) A pointer to a variable in which the test word is written into

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Ram, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_RT_BUILT_IN_TEST_WORD, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalRtBuiltInTestWordSource_MEMORY.

S16BIT _DECL
sitalRt_BuiltInTestWord_Write (S16BIT *swDevice*,
U16BIT *wWord*
)

Write given built-in test word at the appropriate predefined memory address of given RT device.

Equivalent DDC definition: aceRTBITWrdWrite

Parameters:

swDevice (in) Logical number of device (0-31, unused)

wWord (in) The word to write into RT memory as the built-in test word

Returns:

sitalReturnCode_SUCCESS Function successfully completed

Negative sitalReturnCode_* Error condition or function failed

References sitalConfigurationRegister4_RT_EXTERNAL_BUILT_IN_TEST_WORD, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_CONFIGURATION_4, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

```
S16BIT _DECL
sitalRt_DataBlock_Create ( S16BIT    swDevice,
                           S16BIT    swDataBlockId,
                           U16BIT    wDataBlockType,
                           U16BIT *  wapBuffer,
                           U16BIT    wBufferSize
                         )
```

Create for given RT device a data block of given ID and type, and fill it with the contents of given buffer. Use given type to determine both the size of the new data block and whether it will be used by some specific subaddress or globally. In case a common circular data block is created, enter it into use as the common circular buffer of given RT device.

Note:

- Only a single circular data block that's common to all subaddresses of a RT may be created.
- A data block must be allocated at a device memory address that may be divided by its size, so that cyclic wrap will be a matter of zeroing the suitable number of LSBits. In case of a double data block (whose size is 64 = 2 * 32 words), this also causes that each of the two normal data blocks it contains will begin at an address that divides in 32.

Equivalent DDC definition: aceRTDataBlkCreate

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalRtCounter_DATA_BLOCKS-1))
wDataBlockType	(in) The type of data block to create (sitalRtDataBlockType_*)
wapBuffer	(in) A pointer to an array of data words to be copied into the new data block, or NULL if such a copy isn't required
wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (>0)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References
sitalConfigurationRegister6_RT_GLOBAL_CIRCULAR_BUFFER_ENABLE_SIZE_1024,
sitalConfigurationRegister6_RT_GLOBAL_CIRCULAR_BUFFER_ENABLE_SIZE_128,

sitalConfigurationRegister6_RT_GLOBAL_CIRCULAR_BUFFER_ENABLE_SIZE_2048, sitalConfigurationRegister6_RT_GLOBAL_CIRCULAR_BUFFER_ENABLE_SIZE_256, sitalConfigurationRegister6_RT_GLOBAL_CIRCULAR_BUFFER_ENABLE_SIZE_4096, sitalConfigurationRegister6_RT_GLOBAL_CIRCULAR_BUFFER_ENABLE_SIZE_512, sitalConfigurationRegister6_RT_GLOBAL_CIRCULAR_BUFFER_ENABLE_SIZE_8192, sitalConfigurationRegister6_RT_GLOBAL_CIRCULAR_BUFFER_ENABLE_SIZE_MASK, sitalConfigurationRegister6_RT_GLOBAL_CIRCULAR_BUFFER_ENABLE_SIZE_SINGLE, sitalDeviceState_READY, sitalInterruptRegister1_RT_CIRCULAR_BUFFER_ROLLOVER, sitalInterruptRegister2_RT_CIRCULAR_BUFFER_HALF_ROLLOVER, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_CONFIGURATION_6, sitalReturnCode_ALLOCATION_FAIL, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_RT_DATA_BLOCK_EXISTS, sitalReturnCode_SUCCESS, sitalRt_DataBlock_GetSize(), sitalRt_DataBlock_Write, sitalRtDataBlockType_CIRCULAR_1024, sitalRtDataBlockType_CIRCULAR_128, sitalRtDataBlockType_CIRCULAR_2048, sitalRtDataBlockType_CIRCULAR_256, sitalRtDataBlockType_CIRCULAR_4096, sitalRtDataBlockType_CIRCULAR_512, sitalRtDataBlockType_CIRCULAR_8192, sitalRtDataBlockType_COMMON_CIRCULAR_1024, sitalRtDataBlockType_COMMON_CIRCULAR_128, sitalRtDataBlockType_COMMON_CIRCULAR_2048, sitalRtDataBlockType_COMMON_CIRCULAR_256, sitalRtDataBlockType_COMMON_CIRCULAR_4096, sitalRtDataBlockType_COMMON_CIRCULAR_512, sitalRtDataBlockType_COMMON_CIRCULAR_8192, sitalRtDataBlockType_DOUBLE, and sitalWordCountOrModeCodeMask_ALL.

S16BIT _DECL
sitalRt_DataBlock_Delete (S16BIT *swDevice*,
S16BIT *swDataBlockId*
)

Delete given previously created data block of given RT device, and free the segment of device memory that has been allocated for it.

Equivalent DDC definition: aceRTDataBlkDelete

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalRtCounter_DATA_BLOCKS-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, sitalReturnCode_UNDEFINED_NODE, sitalRt_DataBlock_UnmapFromSubaddress, sitalRtMessageType_BROADCAST, sitalRtMessageType_RX, and sitalRtMessageType_TX.

S16BIT _DECL
sitalRt_DataBlock_GetAddress (S16BIT *swDevice*,
S16BIT *swDataBlockId*,
U16BIT * *wpDeviceMemoryAddress*
)

Get the device memory address allocated for given data block of given RT device.

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalRtCounter_DATA_BLOCKS-1))
wpDeviceMemoryAddress	(out) A pointer to the variable in which the address (in words) of device memory allocated for the data block is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalReturnCode_UNDEFINED_NODE.

S16BIT _DECL
sitalRt_DataBlock_GetSize (U16BIT *wDataBlockType*,
U16BIT * *wpSizeOfAllocatedDeviceMemory*,
U16BIT * *wpActualSizeOfDataBlock*
)

Get the actual size of data block and the size it is allocated in device memory for given type of RT data block.

Parameters:

wDataBlockType	(in) The type of data block to create (sitalRtDataBlockType_*)
wpSizeOfAllocatedDeviceMemory	(out) A pointer to the variable in which the size (in words) of device memory allocated for the data block is returned
wpActualSizeOfDataBlock	(out) A pointer to the variable in which the actual size (in words) of the data block is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_SUCCESS, sitalRtDataBlockType_CIRCULAR_1024, sitalRtDataBlockType_CIRCULAR_128, sitalRtDataBlockType_CIRCULAR_2048, sitalRtDataBlockType_CIRCULAR_256, sitalRtDataBlockType_CIRCULAR_4096, sitalRtDataBlockType_CIRCULAR_512, sitalRtDataBlockType_CIRCULAR_8192, sitalRtDataBlockType_COMMON_CIRCULAR_1024, sitalRtDataBlockType_COMMON_CIRCULAR_128, sitalRtDataBlockType_COMMON_CIRCULAR_2048, sitalRtDataBlockType_COMMON_CIRCULAR_256, sitalRtDataBlockType_COMMON_CIRCULAR_4096, sitalRtDataBlockType_COMMON_CIRCULAR_512, sitalRtDataBlockType_COMMON_CIRCULAR_8192, and sitalRtDataBlockType_DOUBLE.

Referenced by sitalRt_DataBlock_Create().

S16BIT _DECL
sitalRt_DataBlock_MapToSubaddress (S16BIT *swDevice*,
S16BIT *swDataBlockId*,
U16BIT *wSubaddress*,
U16BIT *wMessageTypes*,
U16BIT *wIrqOptions*,
U16BIT *bIsSubaddressLegalizationRequested*
)

Map given data block with given subaddress of given RT device for given message types. Enable given interrupts with given subaddress. If required, legalize given message types for:

- All mode code messages
- All messages of given subaddress.

Note:

- Function sitalRt_Initialize illegalizes upon RT initialization messages for all address type, direction, subaddress, and word-count-or-mode-code combinations. To legalize a specific combination, either of the following functions may be used, as appropriate: sitalRt_DataBlock_MapToSubaddress, sitalRt_MessageLegality_Enable.
- Function sitalRt_Initialize illegalizes upon RT initialization messages for all address type, direction, subaddress, and word-count-or-mode-code combinations. To legalize a specific combination, either of the following functions may be used, as appropriate: sitalRt_DataBlock_MapToSubaddress, sitalRt_MessageLegality_Enable.
- Message legalization should be configured using either functions sitalRt_MessageLegality_Enable/sitalRt_MessageLegality_Disable *OR* function sitalRt_DataBlock_MapToSubaddress, not both! Using both to legalize/illegalize messages will disrupt the message legalization lookup taable of the target RT!

Equivalent DDC definition: aceRTDataBlkMapToSA

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalRtCounter_DATA_BLOCKS-1))
wSubaddress	(in) A RT subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wMessageTypes	(in) A set of message types(An or-ed combination of sitalRtMessageType_*)

wIrqOptions	(in) Required interrupts (An or-ed combination of sitalRtDataBlockIrq_*)
bIsSubaddressLegalizationRequested	(in) A flag that says whether given subaddress should be legalized

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Positive integer ID of a data block with which a contradicting mapping already exists, OR given ID (in case required mapping already exists)

References sital1553_MODE_CODE1, sital1553_MODE_CODE2, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMessageDirection_RX, sitalMessageDirection_TX, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_RT_DATA_BLOCK_EXISTS, sitalReturnCode_RT_DATA_BLOCK_MAPPED, sitalReturnCode_SUCCESS, sitalRt_MessageLegality_Enable, sitalRtAddressType_BROADCAST, sitalRtAddressType_OWN, sitalRtCounter_DATA_BLOCKS, sitalRtMessageType_BROADCAST, sitalRtMessageType_RX, sitalRtMessageType_TX, and sitalWordCountOrModeCodeMask_ALL.

```
S16BIT _DECL
sitalRt_DataBlock_Read ( S16BIT    swDevice,
                         S16BIT    swDataBlockId,
                         U16BIT *  wapBuffer,
                         U16BIT    wBufferSize,
                         U16BIT    wOffset
                       )
```

Read the contents of given data block of given RT device starting at given offset into given buffer. For this purpose consider the data block as a cyclic buffer, i.e., continue the copy operation from its start if/when reaching its end.

Equivalent DDC definition: aceRTDataBlkRead

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalRtCounter_DATA_BLOCKS-1))
wapBuffer	(in) A pointer to an array of data words to be copied into the new data block
wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (0-wDeviceMemorySize)
wOffset	(in) The offset (in words) from the beginning of the data block in device memory where cyclic copy begins (0-(<size-of-given-data-block>-1))

Returns:

Positive integer The number of words that were read into given buffer
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalReturnCode_UNDEFINED_NODE.

S16BIT _DECL
sitalRt_DataBlock_UnmapFromSubaddress (S16BIT *swDevice*,
S16BIT *swDataBlockId*,
U16BIT *wSubaddress*,
U16BIT *wMessageTypes*
)

Unmap given data block with given subaddress of given RT device for given message types.

Equivalent DDC definition: aceRTDataBlkUnmapFromSA

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalRtCounter_DATA_BLOCKS-1))
wSubaddress	(in) A RT subaddress (sitalRtSubaddress_1-sitalRtSubaddress_30)
wMessageTypes	(in) A set of message types(An or-ed combination of sitalRtMessageType_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, sitalReturnCode_UNDEFINED_NODE, sitalRtMessageType_BROADCAST, sitalRtMessageType_RX, and sitalRtMessageType_TX.

```
S16BIT _DECL
sitalRt_DataBlock_Write ( S16BIT    swDevice,
                          S16BIT    swDataBlockId,
                          U16BIT *  wapBuffer,
                          U16BIT    wBufferSize,
                          U16BIT    wOffset
                        )
```

Write the contents of given buffer in given data block of given RT device at given offset. For this purpose consider the data block as a cyclic buffer, i.e., continue the copy operation from its start if/when reaching its end.

Equivalent DDC definition: aceRTDataBlkWrite

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
swDataBlockId	(in) A unique ID designating a data block (0-(sitalRtCounter_DATA_BLOCKS-1))
wapBuffer	(in) A pointer to an array of data words to be copied into the new data block
wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (0-wDeviceMemorySize)
wOffset	(in) The offset (in words) from the beginning of the data block in device memory where cyclic copy begins (0-(<size-of-given-data-block>-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalReturnCode_UNDEFINED_NODE.

S16BIT _DECL
sitalRt_HostBuffer_Free (S16BIT *swDevice*)

Free given RT device's host buffer.

Note:

- See the documentation of function sitalRtMt_HostBuffer_Initialize for more information on using functions sitalRt_HostBuffer_* or sitalMt_HostBuffer_* combined with functions sitalRtMt_HostBuffer_*.

Equivalent DDC definition: aceRTUninstallHBuf

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalDeviceState_READY, sitalInterruptRegister1_BC_RT_COMMAND_STACK_ROLLOVER, sitalInterruptRegister1_END_OF_MESSAGE, sitalInterruptRegister1_MT_COMMAND_STACK_ROLLOVER, sitalInterruptRegister1_MT_DATA_STACK_ROLLOVER, sitalInterruptRegister1_TIME_TAG_ROLLOVER, sitalInterruptRegister2_MT_COMMAND_STACK_HALF_ROLLOVER, sitalInterruptRegister2_MT_DATA_STACK_HALF_ROLLOVER, sitalInterruptRegister2_RT_COMMAND_STACK_HALF_ROLLOVER, sitalMode_RT, sitalMode_RT_AND_MT, sitalProcess_SetInterruptServiceRoutine(), sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRt_HostBuffer_GetOperationalStatistics (S16BIT *swDevice*,
sitalHostBufferOperationalStatisticsStructure * *hbosspHostBufferOperationalStatistics*,
U16BIT *bIsResetOfHighestRecordedPercentageRequired*
)

Return performance information about the host buffer of given RT device.

Equivalent DDC definition: aceRTGetHBufMetric

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
hbosspHostBufferOperationalStatistics	(out) A pointer to the host buffer operational statistics structure into which the required operational statistics are written
bIsResetOfHighestRecordedPercentageRequired	(in) A flag that says whether the record of the highest percentage reached by now should be reset

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_OPERATIONAL_STATISTICS_NOT_ENABLED, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRt_HostBuffer_Initialize (S16BIT *swDevice*,
U32BIT *dwHostBufferSize*
)

Initialize (or re-initialize) given RT device's host buffer.

Note:

- Given size of host buffer must be large enough to contain sitalRtMinimum_COMMAND_STACKS_IN_HOST_BUFFER times the number of raw messages that the command stack of given RT device may contain. The number of messages that may be contained by the command stack (N) is its configured size in words divided by sitalRtMemoryObjectSize_COMMAND_STACK_ENTRY. As the maximum possible size of message is allocated for each message in the host buffer, the corresponding size (in words) is (N / sitalRtMaximum_MESSAGE_SIZE). Therefore, given size must be not less than (sitalRtMinimum_COMMAND_STACKS_IN_HOST_BUFFER * (N / sitalRtMaximum_MESSAGE_SIZE)).

Equivalent DDC definition: aceRTInstallHBuf

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dwHostBufferSize	(in) The desired size (in words) of the host buffer (<above-defined-minimum-size-of-RT-host-buffer>-sitalMaximum_SIZE_OF_HOST_BUFFER)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalDeviceState_READY, sitalInterruptRegister1_BC_RT_COMMAND_STACK_ROLLOVER, sitalInterruptRegister1_END_OF_MESSAGE, sitalInterruptRegister1_TIME_TAG_ROLLOVER, sitalInterruptRegister2_RT_COMMAND_STACK_HALF_ROLLOVER, sitalMaximum_SIZE_OF_HOST_BUFFER, sitalMode_RT, sitalMode_RT_AND_MT, sitalProcess_SetInterruptServiceRoutine(), sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_HOST_BUFFER_SIZE, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, sitalRt_HostBuffer_Free, sitalRtMaximum_MESSAGE_SIZE,

sitalRtMemoryObjectSize_COMMAND_STACK_ENTRY, and
sitalRtMinimum_COMMAND_STACKS_IN_HOST_BUFFER.

S16BIT _DECL **sitalRt_HostBuffer_Message_GetCount** (S16BIT *swDevice*)

Get the number of messages that are currently available in the host buffer that is assigned with given RT device.

Equivalent DDC definition: aceRTGetHBufMsgCount

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

Non-negative integer The number of messages that are currently available in the host buffer that is assigned with given device
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, and sitalReturnCode_INVALID_STATE.

S16BIT _DECL **sitalRt_HostBuffer_Message_GetDecoded**	(S16BIT		*swDevice*,
		sitalDecodedMessageStructure *	*dmspDecodedMessage*,
		U32BIT *	*dwpMessageCount*,
		U32BIT *	*dwpStackLostMessageCount*,
		U32BIT *	*dwpHostBufferLostMessageCount*,
		U16BIT	*wMessageLocationAndRemoval*
	)		

Read from the host buffer of given RT device the message at given location, decode it into given structure, and purge it if so required. Also get the number of retrieved messages (actually only 0 or 1), the host buffer's current number of lost messages, and the current number of lost messages for given RT device's RT stack.

Note:

- In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.

Equivalent DDC definition: aceRTGetStkMsgDecoded

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded
dwpMessageCount	(out) A pointer to a variable in which the number of retrieved messages (actually only 0 or 1) is returned
dwpStackLostMessageCount	(out) A pointer to a variable in which given device stack's current number of lost messages is returned
dwpHostBufferLostMessageCount	(out) A pointer to a variable in which the host buffer's current number of lost messages is returned

wMessageLocationAndRemoval	(in) The location in the stack or host buffer of the message to read, and removal instructions (sitalMessageLocationAndRemoval_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMessageLocation_NEXT, sitalMessageRemoval_PURGE, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, sitalRt_Message_DecodeRaw, and sitalRtMaximum_MESSAGE_SIZE.

S16BIT _DECL
sitalRt_HostBuffer_Message_GetRaw (S16BIT *swDevice*,
U16BIT * *wapBuffer*,
U16BIT *wBufferSize*,
U32BIT * *dwpMessageCount*,
U32BIT * *dwpStackLostMessageCount*,
U32BIT * *dwpHostBufferLostMessageCount*
)

Read from the host buffer of given RT device as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the host buffer the messages that were read.

Note:

- Given buffer is first zeroed, and then filled with available messages, where each message is stored as follows:
  - A number of sitalRtMaximum_MESSAGE_SIZE memory words is dedicated per message, never mind its actual size.
  - The target RT stack entry is copied into given buffer.
  - The data stack pointer is replaced with a word whose:
    - MSByte contains the count of data words for the message that was read.
    - LSByte contains the type of the message that was read (sitalMessageType_*).
  - The data words are stored right after the stack entry, that is, starting at offset sitalRtMemoryObjectSize_COMMAND_STACK_ENTRY. In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.

Equivalent DDC definition: aceRTGetHBufMsgsRaw

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wapBuffer	(out) A pointer to a buffer in which the raw messages are stored

wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (>0)
dwpMessageCount	(out) A pointer to a variable in which the number of retrieved messages (>=0) is returned
dwpStackLostMessageCount	(out) A pointer to a variable in which given device stack's current number of lost messages is returned
dwpHostBufferLostMessageCount	(out) A pointer to a variable in which the host buffer's current number of lost messages is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalRtMaximum_MESSAGE_SIZE.

S16BIT _DECL **sitalRt_HostBuffer_Message_Record** (S16BIT *swDevice*)

Record the newly received messages from the command stack of given RT device into its host buffer.

Note:

- User applications should not principally use this function, as it is automatically called by this library whenever a relevant interrupt occurs. This function has been originally exported only in order to enable working with operating systems that does not support interrupts.
- Messages are recorded in the host buffer only as long that it isn't full, that is, there's no override of old messages by new ones. The user must repeatedly get messages from the host buffer in order to empty it.

Equivalent DDC definition: aceRTMTStkToHBuf

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, sitalRtMaximum_MESSAGE_SIZE, and sitalRtMemoryObjectSize_COMMAND_STACK_ENTRY.

S16BIT _DECL
sitalRt_Initialize (S16BIT *swDevice*,
U16BIT *wCommandStackSize*,
U32BIT *dwOptions*
)

Initialize given device as a RT in accordance with given initialization options. Release any past allocations of device memory.

Note:

- This function assumes that function sitalDevice_Initialize has already been called, initialized given device, and inquired its capabilities. It then lets the user reinitialize given device with a non default configuration.
- This function illegalizes messages for all address type, direction, subaddress, and word-count-or-mode-code combinations. To legalize a specific combination, either of the following functions may be used, as appropriate: sitalRt_DataBlock_MapToSubaddress, sitalRt_MessageLegality_Enable.

Equivalent DDC definition: aceRTConfigure

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wCommandStackSize	(in) The desired size (in words) of the command stack (sitalRtCommandStackSize_*)
dwOptions	(in) Initialization options (An or-ed combination of sitalRtSetupOption_*, not including b1553a options)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Zero, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalMode_RT, sitalProcess_SetInterruptServiceRoutine(), sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRt_Message_DecodeRaw (S16BIT *swDevice*,
U16BIT * *wapBuffer*,
sitalDecodedMessageStructure * *dmspDecodedMessage*
)

Decode given message of given RT device into given structure.

Note:

- This function does not really require the logical number of the relevant device, which isn't removed only in order to stay compatible with DDC. (DDC use this parameter to verify the given device is a RT at state READY or RUN, which is a non really required restriction.)
- This function assumes that given buffer is at least sitalRtMaximum_MESSAGE_SIZE words long.
- This function assumes that given buffer has been previously filled by function sitalRt_Message_GetFromStackRaw. See the documentation for this function for information on the contents of its returned buffer.
- In case given buffer contains a mode code message with data, this data word is returned as the first (actually the single) data word of the decoded message.

Equivalent DDC definition: aceRTDecodeRawMsg

Parameters:

swDevice	(in) Logical number of device (0-31, unused)
wapBuffer	(in) A pointer to a buffer in which a raw message is stored
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_SUCCESS, and sitalRtMemoryObjectSize_COMMAND_STACK_ENTRY.

S16BIT _DECL **sitalRt_Message_GetFromStackDecoded**	(S16BIT		*swDevice,*
		sitalDecodedMessageStructure *	*dmspDecodedMessage,*
		U16BIT	*wMessageLocationAndRemoval*
	)		

Read from the stack of given RT device the message at given location, decode it into given structure, and purge it if so required.

Note:

- In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.

Equivalent DDC definition: aceRTGetStkMsgDecoded

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded
wMessageLocationAndRemoval	(in) The location in the stack or host buffer of the message to read, and removal instructions (sitalMessageLocationAndRemoval_*)

Returns:

One (1) The [single] requested message was read and decoded
Zero (0) No message available, though no error occurred
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMessageLocation_NEXT, sitalMessageRemoval_PURGE, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER,

sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, sitalRt_Message_DecodeRaw, and sitalRtMaximum_MESSAGE_SIZE.

S16BIT _DECL
sitalRt_Message_GetFromStackRaw (S16BIT *swDevice*,
U16BIT * *wapBuffer*,
U16BIT *wBufferSize*
)

Read from the RT stack of given RT device into given buffer as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the stack the messages that were read.

Note:

- Given buffer is first zeroed, and then filled with available messages, where each message is stored as follows:
  - A number of sitalRtMaximum_MESSAGE_SIZE memory words is dedicated per message, never mind its actual size.
  - The target RT stack entry is copied into given buffer.
  - The data stack pointer is replaced with a word whose:
    - MSByte contains the count of data words for the message that was read.
    - LSByte contains the type of the message that was read (sitalMessageType_*).
  - The data words are stored right after the stack entry, that is, starting at offset sitalRtMemoryObjectSize_COMMAND_STACK_ENTRY. In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.

Equivalent DDC definition: aceRTGetStkMsgsRaw

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wapBuffer	(out) A pointer to a buffer in which the raw messages are stored
wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (>=sitalRtMaximum_MESSAGE_SIZE)

Returns:

Non-negative integer The number of messages that were read
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalRtMaximum_MESSAGE_SIZE.

S16BIT _DECL
sitalRt_MessageBusyBit_Clear (S16BIT *swDevice*,
U16BIT *wOwnAddressOrBroadcast*,
U16BIT *wMessageDirection*,
U32BIT *dwSubaddressMask*
)

Make messages received by given RT device not raise the busy bit in their status responses if they suit given criteria.

Note:

- The busy lookup table is a 4-double-word area in device memory. It allows any subset of the 128 possible combinations of broadcast / own address, Tx / Rx bit, and subaddress to be configured to have the busy bit set in corresponding responses. It is arranged in a way that the offset of a word from the beginning of the table enables to deduce the related address, direction, and subaddress triplet, as follows:
  - Bit #2 in the binary representation of that offset is 0 / 1 for own address / broadcast, respectively.
  - Bit #1 in the binary representation of that offset is 0 / 1 for receive / transmit, respectively. Then in each double-word entry in the table bit #0...#31 is the busy bit set flag for subaddresses 0...31, respectively..

Equivalent DDC definition: aceRTBusyBitsTblClear

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wOwnAddressOrBroadcast	(in) Message destination selection set (An or-ed combination of sitalRtAddressType_*)
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_*)
dwSubaddressMask	(in) Mask of affected subaddresses, 0-31, where bit #i corresponds subaddress #i (An or-ed combination of sitalRtSubaddressMask_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMessageDirection_BOTH,

sitalMessageDirection_RX, sitalMessageDirection_TX, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_NOT_SUPPORTED, sitalReturnCode_SUCCESS, sitalRtAddressType_BOTH, sitalRtAddressType_BROADCAST, and sitalRtAddressType_OWN.

S16BIT _DECL
sitalRt_MessageBusyBit_GetStatus (S16BIT *swDevice*,
U16BIT *wOwnAddressOrBroadcast*,
U16BIT *wMessageDirection*,
U32BIT * *dwpSubaddressMask*
)

Get the current busy bit state of messages received by given RT device if they suit given criteria.

Note:

- The busy lookup table is a 4-double-word area in device memory. It allows any subset of the 128 possible combinations of broadcast / own address, Tx / Rx bit, and subaddress to be configured to have the busy bit set in corresponding responses. It is arranged in a way that the offset of a word from the beginning of the table enables to deduce the related address, direction, and subaddress triplet, as follows:
  - Bit #2 in the binary representation of that offset is 0 / 1 for own address / broadcast, respectively.
  - Bit #1 in the binary representation of that offset is 0 / 1 for receive / transmit, respectively. Then in each double-word entry in the table bit #0...#31 is the busy bit set flag for subaddresses 0...31, respectively..

Equivalent DDC definition: aceRTBusyBitsTblStatus

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wOwnAddressOrBroadcast	(in) Message destination selection set (sitalRtAddressType_BROADCAST or sitalRtAddressType_OWN)
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX)
dwpSubaddressMask	(out) A pointer to a variable within which a busy bit mask of subaddresses, 0-31, where bit #i corresponds subaddress #i, is returned, what forms an or-ed combination of sitalRtSubaddressMask_*

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_NOT_SUPPORTED, sitalReturnCode_SUCCESS, and sitalRtAddressType_BROADCAST.

S16BIT _DECL
sitalRt_MessageBusyBit_Set (S16BIT *swDevice*,
U16BIT *wOwnAddressOrBroadcast*,
U16BIT *wMessageDirection*,
U32BIT *dwSubaddressMask*
)

Make messages received by given RT device raise the busy bit in their status responses if they suit given criteria.

Note:

- The busy lookup table is a 4-double-word area in device memory. It allows any subset of the 128 possible combinations of broadcast / own address, Tx / Rx bit, and subaddress to be configured to have the busy bit set in corresponding responses. It is arranged in a way that the offset of a word from the beginning of the table enables to deduce the related address, direction, and subaddress triplet, as follows:
  - Bit #2 in the binary representation of that offset is 0 / 1 for own address / broadcast, respectively.
  - Bit #1 in the binary representation of that offset is 0 / 1 for receive / transmit, respectively. Then in each double-word entry in the table bit #0...#31 is the busy bit set flag for subaddresses 0...31, respectively..

Equivalent DDC definition: aceRTBusyBitsTblSet

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wOwnAddressOrBroadcast	(in) Message destination selection set (An or-ed combination of sitalRtAddressType_*)
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_*)
dwSubaddressMask	(in) Mask of affected subaddresses, 0-31, where bit #i corresponds subaddress #i (An or-ed combination of sitalRtSubaddressMask_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMessageDirection_BOTH,

sitalMessageDirection_RX, sitalMessageDirection_TX, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_NOT_SUPPORTED, sitalReturnCode_SUCCESS, sitalRtAddressType_BOTH, sitalRtAddressType_BROADCAST, and sitalRtAddressType_OWN.

S16BIT _DECL
sitalRt_MessageLegality_Disable (S16BIT *swDevice*,
U16BIT *wOwnAddressOrBroadcast*,
U16BIT *wMessageDirection*,
U16BIT *wSubaddress*,
U32BIT *dwWordCountOrModeCodeMask*
)

Illegalize messages received by given RT device if they suit given criteria.

Note:

- The command illegalization table is a 256-word area in device memory. It allows any subset of the 4096 possible combinations of broadcast / own address, Tx / Rx bit, subaddress, and word count / mode code to be illegalized.
- Function sitalRt_Initialize illegalizes upon RT initialization messages for all address type, direction, subaddress, and word-count-or-mode-code combinations. To legalize a specific combination, either of the following functions may be used, as appropriate: sitalRt_DataBlock_MapToSubaddress, sitalRt_MessageLegality_Enable.
- Message legalization should be configured using either functions sitalRt_MessageLegality_Enable/sitalRt_MessageLegality_Disable *OR* function sitalRt_DataBlock_MapToSubaddress, not both! Using both to legalize/illegalize messages will disrupt the message legalization lookup taable of the target RT!

Equivalent DDC definition: aceRTMsgLegalityDisable

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wOwnAddressOrBroadcast	(in) Message destination selection set (An or-ed combination of sitalRtAddressType_*)
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_*)
wSubaddress	(in) Subaddress (sitalRtSubaddress_*)
dwWordCountOrModeCodeMask	(in) Word count or mode code mask (A bitwise mask in which bit-0 / bit-j>0 is on for messages containing 32 / j data words, respectively)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMessageDirection_BOTH, sitalMessageDirection_RX, sitalMessageDirection_TX, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, sitalRtAddressType_BOTH, sitalRtAddressType_BROADCAST, sitalRtAddressType_OWN, and sitalRtSubaddress_ALL.

S16BIT _DECL
sitalRt_MessageLegality_Enable (S16BIT *swDevice*,
U16BIT *wOwnAddressOrBroadcast*,
U16BIT *wMessageDirection*,
U16BIT *wSubaddress*,
U32BIT *dwWordCountOrModeCodeMask*
)

Legalize messages received by given RT device if they suit given criteria.

Note:

- The command illegalization table is a 256-word area in device memory. It allows any subset of the 4096 possible combinations of broadcast / own address, Tx / Rx bit, subaddress, and word count / mode code to be either legalized or illegalized.
- Function sitalRt_Initialize illegalizes upon RT initialization messages for all address type, direction, subaddress, and word-count-or-mode-code combinations. To legalize a specific combination, either of the following functions may be used, as appropriate: sitalRt_DataBlock_MapToSubaddress, sitalRt_MessageLegality_Enable.
- Message legalization should be configured using either functions sitalRt_MessageLegality_Enable/sitalRt_MessageLegality_Disable *OR* function sitalRt_DataBlock_MapToSubaddress, not both! Using both to legalize/illegalize messages will disrupt the message legalization lookup taable of the target RT!

Equivalent DDC definition: aceRTMsgLegalityEnable

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wOwnAddressOrBroadcast	(in) Message destination selection set (An or-ed combination of sitalRtAddressType_*)
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_*)
wSubaddress	(in) Subaddress (sitalRtSubaddress_*)
dwWordCountOrModeCodeMask	(in) Word count or mode code mask (A bitwise mask in which bit-0 / bit-j>0 is on for messages containing 32 / j data words, respectively)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMessageDirection_BOTH, sitalMessageDirection_RX, sitalMessageDirection_TX, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, sitalRtAddressType_BOTH, sitalRtAddressType_BROADCAST, sitalRtAddressType_OWN, and sitalRtSubaddress_ALL.

```
S16BIT _DECL
sitalRt_MessageLegality_GetStatus ( S16BIT     swDevice,
                                    U16BIT     wOwnAddressOrBroadcast,
                                    U16BIT     wMessageDirection,
                                    U16BIT     wSubaddress,
                                    U32BIT *   dwpMessageLegality
                                  )
```

Get the current legalization state of messages received by given RT device if they suit given criteria.

Note:

- The command illegalization table is a 256-word area in device memory. It allows any subset of the 4096 possible combinations of broadcast / own address, Tx / Rx bit, subaddress, and word count / mode code to be illegalized.

Equivalent DDC definition: aceRTMsgLegalityStatus

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wOwnAddressOrBroadcast	(in) Message destination selection set (sitalRtAddressType_BROADCAST or sitalRtAddressType_OWN)
wMessageDirection	(in) RT-related message direction (sitalMessageDirection_RX or sitalMessageDirection_TX)
wSubaddress	(in) Subaddress (0-31)
dwWordCountOrModeCodeMask	(in) Word count or mode code mask (A bitwise mask in which bit-0 / bit-j>0 is on for messages containing 32 / j data words, respectively)
dwpMessageLegality	(out) A pointer to a variable in which the value of the message legality is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN,

sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRt_ModeCode_DisableIrq (S16BIT *swDevice*,
U16BIT *wModeCodeType*,
U16BIT *wModeCodeIrq*
)

Set given RT device to stop the issue of given interrupts upon reception of given mode codes messages.

Note:

- Entries in the mode code selective interrupt table are addressed by appending the 3-bit value formed by broadcast, T/R* bit, and mode code bit 4 to the base address of the mode code selective interrupt table.
- The bit location within the addressed word-entry of the mode code selective interrupt table that determines if the given mode command will generate an interrupt is specified by the 4-bit value formed by mode code bits 3-0.
- An interrupt bit that is set off will cause the device to avoid the generation of an interrupt at the end of the trigerring message.

Equivalent DDC definition: aceRTModeCodeIrqDisable

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wModeCodeType	(in) A type of mode code (sitalRtModeCodeType_*)
wModeCodeIrq	(in) A set of mode code interrupts (An or-ed combination of sitalRtModeCodeIrq_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRt_ModeCode_EnableIrq (S16BIT *swDevice*,
U16BIT *wModeCodeType*,
U16BIT *wModeCodeIrq*
)

Set given RT device to generate given interrupts upon reception of given mode codes messages.

Note:

- Entries in the mode code selective interrupt table are addressed by appending the 3-bit value formed by broadcast, T/R* bit, and mode code bit 4 to the base address of the mode code selective interrupt table.
- The bit location within the addressed word-entry of the mode code selective interrupt table that determines if the given mode command will generate an interrupt is specified by the 4-bit value formed by mode code bits 3-0.
- An interrupt bit that is set on will cause the device to generate an interrupt at the end of the trigerring message.
- Some of the possible command word combinations are invalid. Though, as long as the command word meets the IEEE-1553 criteria for word validation, the interrupt will be generated, even if the particular command is illegal or invalid.

Equivalent DDC definition: aceRTModeCodeIrqEnable

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wModeCodeType	(in) A type of mode code (sitalRtModeCodeType_*)
wModeCodeIrq	(in) A set of mode code interrupts (An or-ed combination of sitalRtModeCodeIrq_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRt_ModeCode_GetIrq (S16BIT *swDevice*,
U16BIT *wModeCodeType*,
U16BIT * *wpModeCodeIrq*
)

Let know whether given RT device is currently configured to generate any interrupts upon reception of given mode codes messages.

Note:

- See the notes for function sitalRt_ModeCode_DisableIrq.

Equivalent DDC definition: aceRTModeCodeIrqStatus

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wModeCodeType	(in) A type of mode code (sitalRtModeCodeType_*)
wpModeCodeIrq	(out) A pointer to a variable in which the currently configured set of mode code interrupts is returned (An or-ed combination of sitalRtModeCodeIrq_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRt_ModeCode_ReadData (S16BIT *swDevice*,
U16BIT *wDataContainingModeCode*,
U16BIT * *wpData*
)

Read the data for given mode code from the mode code data locations table of given RT device.

Note:

- The data word that's received in mode code messages with data is stored both:
  - At the suitable entry of the mode code table.
  - Instead of a data pointer at the dedicated entry in the command stack.

Equivalent DDC definition: aceRTModeCodeReadData

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wDataContainingModeCode	(in) A data containing mode code (>=0, <sitalRtCounter_MODE_CODES)
wpData	(out) A pointer to a variable in which the data is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalRtCounter_MODE_CODES.

S16BIT _DECL
sitalRt_ModeCode_WriteData (S16BIT *swDevice*,
U16BIT *wDataContainingModeCode*,
U16BIT *wData*
)

Write given data for given mode code in the mode code data locations table of given RT device.

Equivalent DDC definition: aceRTModeCodeWriteData

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wDataContainingModeCode	(in) A data-containing mode code (>=0, <sitalRtCounter_MODE_CODES)
wData	(in) The data word to write

Returns:

sitalReturnCode_SUCCESS	Function successfully completed
Negative sitalReturnCode_*	Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Write, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalRtCounter_MODE_CODES.

S16BIT _DECL
sitalRt_ResponseStatusBits_Get (S16BIT *swDevice*,
U16BIT * *wpStatusEnablerMask*
)

Get the current configuration of status enabler bits, which are the configuration bits that control status responses made by given RT device. Whether given RT device is configured with/without alternate status mode affects the set of the relevant status bits.

Note:

- An interrupt is issued whenever an interrupt condition occurs, if both the respective enabler bit and the respective bit of the interrupt mask register are set.

Equivalent DDC definition: aceRTStatusBitsStatus

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wpStatusEnablerMask	(out) A pointer to a variable within which the status enabler bit mask in configuration register #1 are returned as an or-ed combination of sitalRtStatusWordBit_*

Returns:

sitalReturnCode_SUCCESS	Function successfully completed
Negative sitalReturnCode_*	Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Read, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_CONFIGURATION_1, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRt_ResponseStatusBits_Set (S16BIT *swDevice*,
U16BIT *wStatusEnablerMask*
)

Configure given status enabler bits in order to control status responses made by given RT device. In case given RT device has been configured to be with/without alternate status mode, given bits are set/cleared, respectively. This configuration of alternate status mode also affects the set of the relevant status bits.

Note:

- An interrupt is issued whenever an interrupt condition occurs, if both the respective enabler bit and the respective bit of the interrupt mask register are set.
- Only bits of given wStatusEnablerMask that really serve as status enabler bits in configuration register #1 are taken in account. Irrelevant bits are ignored.

Equivalent DDC definition: aceRTStatusBitsSet

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wStatusEnablerMask	(in) Status enabler bit mask to update in configuration register #1 with (An or-ed combination of sitalRtStatusWordBit_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_CONFIGURATION_1, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRt_ResponseStatusBits_Unset (S16BIT *swDevice*,
U16BIT *wStatusEnablerMask*
)

Reverse configure given status enabler bits in order to control status responses made by given RT device. In case given RT device has been configured to be with/without alternate status mode, given bits are cleared/set, respectively. This configuration of alternate status mode also affects the set of the relevant status bits.

Note:

- An interrupt is issued whenever an interrupt condition occurs, if both the respective enabler bit and the respective bit of the interrupt mask register are set.
- Only bits of given wStatusEnablerMask that really serve as status enabler bits in configuration register #1 are taken in account. Irrelevant bits are ignored.

Equivalent DDC definition: aceRTStatusBitsClear

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wStatusEnablerMask	(in) Status enabler bit mask to update in configuration register #1 with (An or-ed combination of sitalRtStatusWordBit_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDevice_AccessMemory(), sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalMode_RT_AND_MT, sitalRegisterAddress_CONFIGURATION_1, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL **sitalRt_Start** (S16BIT *swDevice*)

Make given RT device start receiving messages.

Equivalent DDC definition: aceRTStart

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalAccess_USER, sitalConfigurationRegister1_FUNCTION_MASK, sitalConfigurationRegister1_FUNCTION_RT, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalRegisterAddress_CONFIGURATION_1, sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL sitalRt_Stop (S16BIT *swDevice*)

Stop given RT device from responding to received messages.

Equivalent DDC definition: aceRTStop

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalAccess_USER, sitalConfigurationRegister1_FUNCTION_MT, sitalConfigurationRegister1_FUNCTION_RT, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT, sitalRegisterAddress_CONFIGURATION_1, sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRtMt_HostBuffer_Free (S16BIT *swDevice*)

Free given device's RT&MT combined host buffer.

Note:

- See the documentation of function sitalRtMt_HostBuffer_Initialize for more information on using functions sitalRt_HostBuffer_* or sitalMt_HostBuffer_* combined with functions sitalRtMt_HostBuffer_*.

Equivalent DDC definition: aceRTMTUninstallHBuf

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalDeviceState_READY, sitalInterruptRegister1_BC_RT_COMMAND_STACK_ROLLOVER, sitalInterruptRegister1_END_OF_MESSAGE, sitalInterruptRegister1_MT_COMMAND_STACK_ROLLOVER, sitalInterruptRegister1_MT_DATA_STACK_ROLLOVER, sitalInterruptRegister1_TIME_TAG_ROLLOVER, sitalInterruptRegister2_MT_COMMAND_STACK_HALF_ROLLOVER, sitalInterruptRegister2_MT_DATA_STACK_HALF_ROLLOVER, sitalInterruptRegister2_RT_COMMAND_STACK_HALF_ROLLOVER, sitalMode_RT_AND_MT, sitalProcess_SetInterruptServiceRoutine(), sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL **sitalRtMt_HostBuffer_GetOperationalStatistics**	(S16BIT		*swDevice,*
		sitalHostBufferOperationalStatisticsStructure *	*hbosspHostBufferOperationalStatistics,*
		U16BIT	*bIsResetOfHighestRecordedPercentageRequired*
	)		

Return performance information about the combined host buffer of given RT&MT device.

Equivalent DDC definition: aceRTMTGetHBufMetric

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
hbosspHostBufferOperationalStatistics	(out) A pointer to the host buffer operational statistics structure into which the required operational statistics are written
bIsResetOfHighestRecordedPercentageRequired	(in) A flag that says whether the record of the highest percentage reached by now should be reset

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT_AND_MT, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_OPERATIONAL_STATISTICS_NOT_ENABLED, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRtMt_HostBuffer_Initialize (S16BIT *swDevice*,
U32BIT *dwHostBufferSize*
)

Initialize (or re-initialize) given device's RT&MT combined host buffer.

Note:

- The RT&MT combined host buffer is actually a combination of two separate host buffers, RT host buffer and MT host buffer, each of which manipulated exactly like when used by regular RT or MT devices, respectively. No additional or temporary host buffers are used. In this mode of operation, whenever relevant, functions sitalRtMt_HostBuffer_* search the RT and MT host buffers to find the earliest available message, and fetch it. In order to use this mode of operation, the user application must specify option sitalRtMtSetupOption_COMBINED_HOST_BUFFER when calling function sitalRtMt_Initialize. Yet, even in case it specified this option, the caller user application, the user may prefer to use functions sitalRt_HostBuffer_* or sitalMt_HostBuffer_* over functions sitalRtMt_HostBuffer_*, or somehow combine these functions. In the last case, the user application must be fully aware of the state of the RT and MT host buffers; for example:
    - After getting a message using function sitalMt_HostBuffer_Message_GetRaw, this message may not, of course, be read again using function sitalRtMt_HostBuffer_Message_GetRaw.
    - After freeing the MT host buffer using function sitalMt_HostBuffer_Free, no MT messages will be, of course, read by further calls to function sitalRtMt_HostBuffer_Message_GetRaw.
- Given size of host buffer must be large enough to contain sitalRtMinimum_COMMAND_STACKS_IN_HOST_BUFFER times the number of raw messages that the command stack of given RT device may contain. The number of messages that may be contained by the command stack (N) is its configured size in words divided by sitalRtMemoryObjectSize_COMMAND_STACK_ENTRY. As the maximum possible size of message is allocated for each message in the host buffer, the corresponding size (in words) is (N / sitalRtMaximum_MESSAGE_SIZE). Therefore, given size must be not less than (sitalRtMinimum_COMMAND_STACKS_IN_HOST_BUFFER * (N / sitalRtMaximum_MESSAGE_SIZE)).
- Given size of host buffer must be large enough to contain sitalMtMinimum_COMMAND_STACKS_IN_HOST_BUFFER times the number of raw messages that the command stack of given MT device may contain. The number of messages that may be contained by the command stack (N) is its configured size in words divided by sitalMtMemoryObjectSize_COMMAND_STACK_ENTRY. As the maximum

possible size of message is allocated for each message in the host buffer, the corresponding size (in words) is (N / sitalMtMaximum_MESSAGE_SIZE). Therefore, given size must be not less than (sitalMtMinimum_COMMAND_STACKS_IN_HOST_BUFFER * (N / sitalMtMaximum_MESSAGE_SIZE)).

Equivalent DDC definition: aceRTMTInstallHBuf

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dwHostBufferSize	(in) The desired size (in words) of each of the RT and MT host buffer (<above-defined-minimum-size-of-RT&MT-host-buffer>-sitalMaximum_SIZE_OF_HOST_BUFFER)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalDeviceState_READY, sitalInterruptRegister1_BC_RT_COMMAND_STACK_ROLLOVER, sitalInterruptRegister1_END_OF_MESSAGE, sitalInterruptRegister1_MT_COMMAND_STACK_ROLLOVER, sitalInterruptRegister1_MT_DATA_STACK_ROLLOVER, sitalInterruptRegister1_TIME_TAG_ROLLOVER, sitalInterruptRegister2_MT_COMMAND_STACK_HALF_ROLLOVER, sitalInterruptRegister2_MT_DATA_STACK_HALF_ROLLOVER, sitalInterruptRegister2_RT_COMMAND_STACK_HALF_ROLLOVER, sitalMaximum_SIZE_OF_HOST_BUFFER, sitalMode_RT_AND_MT, sitalMt_HostBuffer_Free, sitalMtMaximum_MESSAGE_SIZE, sitalMtMemoryObjectSize_COMMAND_STACK_ENTRY, sitalMtMinimum_COMMAND_STACKS_IN_HOST_BUFFER, sitalProcess_SetInterruptServiceRoutine(), sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_HOST_BUFFER_SIZE, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_RT_AND_MT_NON_COMBINED_HOST_BUFFER_MODE, sitalReturnCode_SUCCESS, sitalRt_HostBuffer_Free, sitalRtMaximum_MESSAGE_SIZE, sitalRtMemoryObjectSize_COMMAND_STACK_ENTRY, sitalRtMinimum_COMMAND_STACKS_IN_HOST_BUFFER, and sitalRtMtSetupOption_COMBINED_HOST_BUFFER.

S16BIT _DECL
sitalRtMt_HostBuffer_Message_GetCount (S16BIT *swDevice*)

Get the number of messages that are currently available in the combined host buffer that is assigned with given RT&MT device.

Note:

- See the documentation of function sitalRtMt_HostBuffer_Initialize for more information on using functions sitalRt_HostBuffer_* or sitalMt_HostBuffer_* combined with functions sitalRtMt_HostBuffer_*.

Equivalent DDC definition: aceRTMTGetHBufMsgCount

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

Non-negative integer The number of messages that are currently available in the host buffer that is assigned with given device
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT_AND_MT, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, and sitalReturnCode_INVALID_STATE.

S16BIT _DECL
sitalRtMt_HostBuffer_Message_GetDecoded (S16BIT *swDevice*,
sitalDecodedMessageStructure * *dmspDecodedMessage*,
U32BIT * *dwpMessageCount*,
U32BIT * *dwpRtStackLostMessageCount*,
U32BIT * *dwpMtStackLostMessageCount*,
U32BIT * *dwpHostBufferLostMessageCount*,
U16BIT *wMessageLocationAndRemoval*
)

Read from the combined host buffer of given RT&MT device the message at given location, decode it into given structure, and purge it if so required. Also get the number of retrieved messages (actually only 0 or 1), the combined host buffer's current number of lost messages, the current number of lost messages for given RT&MT device's RT stack, and the current number of lost messages for both given RT&MT device's MT stacks.

Note:

- In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.
- See the documentation of function sitalRtMt_HostBuffer_Initialize for more information on using functions sitalRt_HostBuffer_* or sitalMt_HostBuffer_* combined with functions sitalRtMt_HostBuffer_*.

Equivalent DDC definition: aceRTMTGetStkMsgDecoded

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
dmspDecodedMessage	(out) A pointer to a structure into which a message is decoded

dwpMessageCount	(out) A pointer to a variable in which the number of retrieved messages (actually only 0 or 1) is returned
dwpRtStackLostMessageCount	(out) A pointer to a variable in which the current number of lost messages for given RT&MT device's RT stack is returned
dwpMtStackLostMessageCount	(out) A pointer to a variable in which the current number of lost messages for both given RT&MT device's MT stacks is returned
dwpHostBufferLostMessageCount	(out) A pointer to a variable in which the combined host buffer's current number of lost messages is returned
wMessageLocationAndRemoval	(in) The location in the stack or host buffer of the message to read, and removal instructions (sitalMessageLocationAndRemoval_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT_AND_MT, sitalMt_HostBuffer_Message_GetDecoded, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, sitalReturnCode_UNKNOWN_ERROR, and sitalRt_HostBuffer_Message_GetDecoded.

EXTERN S16BIT _DECL
sitalRtMt_HostBuffer_Message_GetRaw (S16BIT *swDevice*,
U16BIT * *wapBuffer*,
U16BIT *wBufferSize*,
U32BIT * *dwpMessageCount*,
U32BIT * *dwpRtStackLostMessageCount*,
U32BIT * *dwpMtStackLostMessageCount*,
U32BIT * *dwpHostBufferLostMessageCount*
)

Read from the combined host buffer of given RT&MT device as many currently available raw messages as possible (that is, as the size of given buffer permits). Remove from the combined host buffer the messages that were read.

Note:

- Given buffer is first zeroed, and then filled with available messages, where each message is stored as follows:
  - A number of sitalRtMtMaximum_MESSAGE_SIZE memory words is dedicated per message, never mind its actual size.
  - The target RT&MT stack entry is copied into given buffer.
  - The data stack pointer is replaced with a word whose:
    - MSByte contains the count of data words for the message that was read.
    - LSByte contains the type of the message that was read (sitalMessageType_*).
  - The data words are stored right after the stack entry, that is, starting at offset sitalRtMtMemoryObjectSize_COMMAND_STACK_ENTRY. In case of a mode code message with data, this data word is returned as the first (actually the single) data word.
- In case a host buffer is assigned to a device, then right after any new message is recorded in this host buffer, that message is purged from the stack. In other words, the usage of a host buffer and the usage of sital*_Message_Get* functions is basically mutually exclussive.
- See the documentation of function sitalRtMt_HostBuffer_Initialize for more information on using functions sitalRt_HostBuffer_* or sitalMt_HostBuffer_* combined with functions sitalRtMt_HostBuffer_*.

Equivalent DDC definition: aceRTMTGetHBufMsgsRaw

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wapBuffer	(out) A pointer to a buffer in which the raw messages are stored
wBufferSize	(in) The size (in words) of the data buffer that is pointed by wapBuffer (>0)
dwpMessageCount	(out) A pointer to a variable in which the number of retrieved messages (>=0) is returned
dwpRtStackLostMessageCount	(out) A pointer to a variable in which the current number of lost messages for given RT&MT device's RT stack is returned
dwpMtStackLostMessageCount	(out) A pointer to a variable in which the current number of lost messages for both given RT&MT device's MT stacks is returned
dwpHostBufferLostMessageCount	(out) A pointer to a variable in which the combined host buffer's current number of lost messages is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMessageLocationAndRemoval_NEXT_PURGE, sitalMode_RT_AND_MT, sitalMt_HostBuffer_Message_GetRaw, sitalMtMaximum_MESSAGE_SIZE, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, sitalReturnCode_UNKNOWN_ERROR, sitalRt_HostBuffer_Message_GetRaw, sitalRtMaximum_MESSAGE_SIZE, and sitalRtMtMaximum_MESSAGE_SIZE.

S16BIT _DECL
sitalRtMt_HostBuffer_Message_Record (S16BIT *swDevice*)

Record the newly received messages from the active command and data stacks of given RT&MT device into its combined host buffer.

Note:

- User applications should not principally use this function, as it is automatically called by this library whenever a relevant interrupt occurs. This function has been originally exported only in order to enable working with operating systems that does not support interrupts.
- Messages are recorded in the host buffer only as long that it isn't full, that is, there's no override of old messages by new ones. The user must repeatedly get messages from the host buffer in order to empty it.
- See the documentation of function sitalRtMt_HostBuffer_Initialize for more information on using functions sitalRt_HostBuffer_* or sitalMt_HostBuffer_* combined with functions sitalRtMt_HostBuffer_*.

Equivalent DDC definition: aceRTMTStkToHBuf

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT_AND_MT, sitalMt_HostBuffer_Message_Record, sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, sitalReturnCode_SUCCESS, and sitalRt_HostBuffer_Message_Record.

```
S16BIT _DECL
sitalRtMt_Initialize ( S16BIT  swDevice,
                       U16BIT  wRtCommandStackSize,
                       U16BIT  wMtStackMode,
                       U16BIT  wMtCommandStackSize,
                       U16BIT  wMtDataStackSize,
                       U32BIT  dwOptions
                     )
```

Initialize given device as a RT&MT in accordance with given initialization options. Release any past allocations of device memory.

Note:

- This function assumes that function sitalDevice_Initialize has already been called, initialized given device, and inquired its capabilities. This function then lets the user reinitialize given device with a non default configuration.
- This function illegalizes messages for all address type, direction, subaddress, and word-count-or-mode-code combinations. To legalize a specific combination, either of the following functions may be used, as appropriate: sitalRt_DataBlock_MapToSubaddress, sitalRt_MessageLegality_Enable.
- Both MT stacks, command and data, must be aligned in the device memory to their size (e.g., a 4096 word stack may be placed only at an address of the form 4096xN where N=0,1,2,...).
- In case the caller specifies stack sizes that the memory of given device is too small to contain, the function call will be rejected.

Equivalent DDC definition: aceRTMTConfigure

Parameters:

swDevice	(in) Logical number of device (0-(sitalMaximum_DEVICES-1))
wRtCommandStackSize	(in) The desired size (in words) of the RT command stack (sitalRtCommandStackSize_*)
wMtStackMode	(in) MT stack mode (principally sitalMtStackOption_*, actually limited to sitalMtStackOption_SINGLE)
wMtCommandStackSize	(in) Size of MT command stack (sitalMtCommandStackSize_*)
wMtDataStackSize	(in) Size of MT data stack (sitalMtDataStackSize_*)

dwOptions	(in) Initialization options (An or-ed combination of sitalRtSetupOption_*, sitalMtSetupOption_*, and sitalRtMtSetupOption_*)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_Zero, sitalDeviceMemorySection_Ram, sitalDeviceState_READY, sitalMode_RT_AND_MT, sitalMtStackOption_SINGLE, sitalProcess_SetInterruptServiceRoutine(), sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_PARAMETER, sitalReturnCode_INVALID_STATE, sitalReturnCode_MT_INVALID_COMMAND_STACK_SIZE, sitalReturnCode_MT_INVALID_DATA_STACK_SIZE, sitalReturnCode_MT_INVALID_STACK_OPTION, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRtMt_Start (S16BIT *swDevice*)

Make given RT&MT device start responding and capturing messages.

Equivalent DDC definition: aceRTMTStart

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalAccess_USER, sitalConfigurationRegister1_FUNCTION_RT, sitalConfigurationRegister1_RT_MESSAGE_MONITOR_ENABLED, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT_AND_MT, sitalRegisterAddress_CONFIGURATION_1, sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalRtMt_Stop (S16BIT *swDevice*)

Stop given RT&MT device from responding to received messages.

Equivalent DDC definition: aceRTMTStop

Parameters:

swDevice (in) Logical number of device (0-(sitalMaximum_DEVICES-1))

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalAccess_CARD, sitalAccess_USER, sitalConfigurationRegister1_FUNCTION_MT, sitalConfigurationRegister1_FUNCTION_RT, sitalDevice_AccessMemory(), sitalDeviceAccessOperation_WriteMasked, sitalDeviceMemorySection_Registers, sitalDeviceState_READY, sitalDeviceState_RUN, sitalMode_RT_AND_MT, sitalRegisterAddress_CONFIGURATION_1, sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_MODE, sitalReturnCode_INVALID_STATE, and sitalReturnCode_SUCCESS.

S16BIT _DECL
sitalStld1553_ConfigurePrintings (BOOLEAN *bIsConsoleScreen*,
sitalPrintingsLevelEnum *pleBasePrintingsLevel*
)

Configure the printings to the console screen according to given parameters. This configuration affects this library as well as the driver-interface library. No printings are made if the current user application has no console screen. Only printings of a level greater/equal to given base printings level will be made.

Parameters:

bIsConsoleScreen	(in) A flag that says whether the current user application has a console screen at all
pleBasePrintingsLevel	(in) The desired base level of printings (sitalPrintingsLevel_*)

Returns:

sitalReturnCode_SUCCESS	Function successfully completed
Negative sitalReturnCode_*	Error condition or function failed

References sitalDriverInterfaceLibrary_ConfigurePrintings().

S8BIT* _DECL sitalStld1553_GetBlockStatusWordErrorString (U16BIT *wMode*,
U16BIT *wBlockStatus*
)

Build and return a string in which the designated errors are textually reported. If no error is designated, a null string is returned.

Note:

- Block status word errors are relevant only with modes BC, RT, and MT, and the interpretation of errors is mode-dependent.
- This function returns the same error string as the library of DDC, just without the redundant trailing space, in order to stay compatible with DDC.
- This function returns a pointer to a static string and not to a dynamically allocated string or a user supplied string only in order to stay compatible with DDC. This behavior isn't thread safe, and in case thread-1 calls this function right after thread-0, the string that has been originally returned to thread-0 is principally changed.

Equivalent DDC definition: aceGetBSWErrString

Parameters:

wMode	(in) Operation mode (sitalMode_*)
wBlockStatus	(in) A block status word that typically designates one or more errors (An or-ed combination of sital*BlockStatusWord_*)

Returns:

A pointer to a string in which the designated errors are textually reported (an empty string is returned in case an impossible mode/status combination is given)

References sitalBcBlockStatusWord_BAD_SYNCHRONIZATION, sitalBcBlockStatusWord_ERROR_FLAG, sitalBcBlockStatusWord_FORMAT_ERROR, sitalBcBlockStatusWord_INVALID_WORD, sitalBcBlockStatusWord_LOOPBACK_FAIL, sitalBcBlockStatusWord_NO_RESPONSE, sitalBcBlockStatusWord_WORD_COUNT_ERROR, sitalBcBlockStatusWord_WRONG_RT_ADDRESS, sitalMode_BC, sitalMode_MT, sitalMode_RT, sitalMtBlockStatusWord_BAD_SYNCHRONIZATION, sitalMtBlockStatusWord_COMMAND_ERROR, sitalMtBlockStatusWord_ERROR_FLAG, sitalMtBlockStatusWord_INVALID_WORD, sitalMtBlockStatusWord_NO_RESPONSE, sitalMtBlockStatusWord_PROTOCOL_VIOLATION, sitalMtBlockStatusWord_RT_TO_RT_COMMAND_ERROR, sitalMtBlockStatusWord_RT_TO_RT_RESPONSE_ERROR, sitalMtBlockStatusWord_WORD_COUNT_ERROR, sitalRtBlockStatusWord_BAD_SYNCHRONIZATION, sitalRtBlockStatusWord_COMMAND_ERROR, sitalRtBlockStatusWord_ERROR_FLAG, sitalRtBlockStatusWord_FORMAT_ERROR, sitalRtBlockStatusWord_ILLEGAL_COMMAND, sitalRtBlockStatusWord_INVALID_WORD, sitalRtBlockStatusWord_LOOPBACK_FAIL, sitalRtBlockStatusWord_NO_RESPONSE, sitalRtBlockStatusWord_RT_TO_RT_COMMAND_ERROR, sitalRtBlockStatusWord_RT_TO_RT_RESPONSE_ERROR, and sitalRtBlockStatusWord_WORD_COUNT_ERROR.

S16BIT _DECL
sitalStld1553_GetLibraryVersion (U16BIT * *wpMajorVersion*,
U16BIT * *wpMinorVersion*,
U16BIT * *wpBuildNumber*,
U16BIT * *wpRevisionNumber*
)

Get this library version numbers.

Parameters:

wpMajorVersion	(out) A pointer to the variable in which the first version number is returned
wpMinorVersion	(out) A pointer to the variable in which the second version number is returned
wpBuildNumber	(out) A pointer to the variable in which the third version number is returned
wpRevisionNumber	(out) A pointer to the variable in which the fourth version number is returned

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

Referenced by sitalStld1553_GetShortLibraryVersion().

S8BIT* _DECL sitalStld1553_GetMessageTypeString (U16BIT *wMessageType*)

Build and return a text string describing given message type.

Note:

- This function returns a pointer to a static string and not to a dynamically allocated string or a user supplied string only in order to stay compatible with DDC. This behavior isn't thread safe, and in case thread-1 calls this function right after thread-0, the string that has been originally returned to thread-0 is principally changed.

Equivalent DDC definition: aceGetMsgTypeString

Parameters:

wMessageType (in) Operation mode (sitalMessageType_*)

Returns:

A pointer to a string in which the designated errors are textually reported (an empty string is returned in case an impossible mode/status combination is given)

S16BIT _DECL
sitalStld1553_GetReturnCodeDescriberString (S16BIT *swError*,
S8BIT * *szpErrorString*,
U16BIT *wMaximumStringLength*
)

Return a string that describes given return code.

Note:

- Given return code may be either an error code, a warning code, or a notification code.
- If given return code code isn't recognized, a suitable notification string is returned, and the function reports success.

Equivalent DDC definition: aceErrorStr

Parameters:

swError	(in) Error code (sitalReturnCode_*)
szpErrorString	(out) A pointer to a string buffer in which a null-terminated string that describes given error code is returned
wMaximumStringLength	(in) The size of the buffer in which the error string is returned, including the terminating null character (>=80)

Returns:

sitalReturnCode_SUCCESS Function successfully completed
Negative sitalReturnCode_* Error condition or function failed

References sitalReturnCode_ALLOCATION_FAIL,
sitalReturnCode_ASYNCHRONOUS_LIST_IS_EMPTY,
sitalReturnCode_ASYNCHRONOUS_LIST_NOT_EMPTY,
sitalReturnCode_ASYNCHRONOUS_MESSAGE_ERROR,
sitalReturnCode_ASYNCHRONOUS_SUBROUTINE_BUSY,
sitalReturnCode_BC_DATA_BLOCK_ALLOCATION_FAIL,
sitalReturnCode_BC_INVALID_DATA_BLOCK_SIZE,
sitalReturnCode_BC_OBJECT_ALREADY_EXISTS,
sitalReturnCode_BC_POSSIBLY_DETRIMENTAL_OPCODE,
sitalReturnCode_DEVICE_CONNECTION_FAIL, sitalReturnCode_DRIVER_OPEN_FAIL,
sitalReturnCode_FRAME_NESTING_RECURSION,
sitalReturnCode_FRAME_NESTING_STACK_OVERFLOW,
sitalReturnCode_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_ILLEGAL_FRAME,
sitalReturnCode_INVALID_ACCESS, sitalReturnCode_INVALID_ADDRESS,
sitalReturnCode_INVALID_ADDRESS_MODE, sitalReturnCode_INVALID_ALLOCATION,

sitalReturnCode_INVALID_BUFFER, sitalReturnCode_INVALID_CAN_ADDRESS,
sitalReturnCode_INVALID_CARD, sitalReturnCode_INVALID_CARD_NUMBER,
sitalReturnCode_INVALID_CLOCK_FREQUENCY,
sitalReturnCode_INVALID_DEVICE_NUMBER, sitalReturnCode_INVALID_DIO_PORT,
sitalReturnCode_INVALID_DIRECTION_BIT, sitalReturnCode_INVALID_FILE,
sitalReturnCode_INVALID_HOST_BUFFER_SIZE,
sitalReturnCode_INVALID_MEMORY_SIZE,
sitalReturnCode_INVALID_MESSAGE_OPTIONS,
sitalReturnCode_INVALID_MESSAGE_STRUCTURE,
sitalReturnCode_INVALID_MESSAGE_TYPE, sitalReturnCode_INVALID_MODE,
sitalReturnCode_INVALID_MODE_OPTIONS,
sitalReturnCode_INVALID_OPERATING_SYSTEM,
sitalReturnCode_INVALID_PARAMETER,
sitalReturnCode_INVALID_REGISTER_ADDRESS,
sitalReturnCode_INVALID_RESPONSE_TIMEOUT,
sitalReturnCode_INVALID_RT_ADDRESS, sitalReturnCode_INVALID_STATE,
sitalReturnCode_INVALID_SUBADDRESS_OR_MODE_SELECTOR,
sitalReturnCode_INVALID_TEST_STRUCTURE,
sitalReturnCode_INVALID_TIME_TAG_RESOLUTION,
sitalReturnCode_INVALID_WORD_COUNT_OR_MODE_CODE,
sitalReturnCode_ISQ_DISABLED, sitalReturnCode_LIBRARY_LOAD_ERROR,
sitalReturnCode_MAPPED_MEMORY_ACCESS_FAIL,
sitalReturnCode_MEMORY_MAP_FAIL, sitalReturnCode_MESSAGE_ERROR,
sitalReturnCode_MESSAGE_NOT_DETECTED,
sitalReturnCode_MT_HOST_BUFFER_NOT_INSTALLED,
sitalReturnCode_MT_INVALID_COMMAND_STACK_SIZE,
sitalReturnCode_MT_INVALID_DATA_STACK_SIZE,
sitalReturnCode_MT_INVALID_DIRECTION_BIT,
sitalReturnCode_MT_INVALID_HOST_BUFFER_SIZE,
sitalReturnCode_MT_INVALID_MESSAGE_LOCATION,
sitalReturnCode_MT_INVALID_RT_ADDRESS,
sitalReturnCode_MT_INVALID_STACK_OPTION,
sitalReturnCode_MT_INVALID_STACK_SELECTOR,
sitalReturnCode_MT_INVALID_SUBADDRESS_BUFFER,
sitalReturnCode_NESTED_MAJOR_FRAME, sitalReturnCode_NODE_EXISTS,
sitalReturnCode_NODE_NOT_MEMORY_BLOCK, sitalReturnCode_NOT_MAJOR_FRAME,
sitalReturnCode_NOT_SUPPORTED, sitalReturnCode_OPERATION_FAIL,
sitalReturnCode_OPERATIONAL_STATISTICS_NOT_ENABLED,
sitalReturnCode_POSSIBLY_DETRIMENTAL_CONFIGURATION,
sitalReturnCode_READ_ERROR, sitalReturnCode_REGISTERS_ACCESS_FAIL,
sitalReturnCode_RT_AND_MT_HOST_BUFFER_NOT_INSTALLED,
sitalReturnCode_RT_AND_MT_INVALID_HOST_BUFFER_SIZE,
sitalReturnCode_RT_AND_MT_INVALID_MESSAGE_LOCATION,

sitalReturnCode_RT_AND_MT_NON_COMBINED_HOST_BUFFER_MODE,
sitalReturnCode_RT_DATA_BLOCK_ALLOCATION_FAIL,
sitalReturnCode_RT_DATA_BLOCK_EXISTS,
sitalReturnCode_RT_DATA_BLOCK_MAPPED,
sitalReturnCode_RT_DATA_BLOCK_NOT_CIRCULAR,
sitalReturnCode_RT_HOST_BUFFER_NOT_INSTALLED, sitalReturnCode_SUCCESS,
sitalReturnCode_SUITABLE_ASYNCHRONOUS_MODE_UNDEFINED,
sitalReturnCode_TASK_SPAWN_FAIL, sitalReturnCode_TEST_FAIL,
sitalReturnCode_TOO_MANY_DEVICES, sitalReturnCode_UNDEFINED_COMMAND,
sitalReturnCode_UNDEFINED_DATA_BLOCK,
sitalReturnCode_UNDEFINED_MESSAGE_BLOCK, sitalReturnCode_UNDEFINED_NODE,
sitalReturnCode_UNKNOWN_ERROR,
sitalReturnCode_UNRESOLVED_ASYNCHRONOUS_COMMAND,
sitalReturnCode_UNRESOLVED_JUMP, sitalReturnCode_VERSION_ERROR, and
sitalReturnCode_WRITE_ERROR.

U16BIT _DECL
sitalStld1553_GetShortCoreVersion (void)

Get the device driver interface library version numbers in a short form.

Equivalent DDC definition: aceGetCoreVersion

Returns:

Positive integer An unsigned 16-bit value, where the MSByte contains the major version number, the MSNibble of the LSByte contains the minor version number, and the LSNibble of the LSByte contains the build number
Zero Error condition or function failed

References sitalDriverInterfaceLibrary_GetVersion(), and sitalReturnCode_SUCCESS.

U16BIT _DECL
sitalStld1553_GetShortLibraryVersion (void)

Get this library version numbers in a short form.

Equivalent DDC definition: aceGetLibVersion

Returns:

Positive integer An unsigned 16-bit value, where the MSByte contains the major version number, the MSNibble of the LSByte contains the minor version number, and the LSNibble of the LSByte contains the build number
Zero Error condition or function failed

References sitalReturnCode_SUCCESS, and sitalStld1553_GetLibraryVersion().